



[illegible]

```
0001 0 module uafmain (main = start,  
0002 0     language (bliss32),  
0003 0     ident = 'v04-000',  
0004 0     addressing_mode (external=general, nonexternal=general)  
0005 0 ) =  
0006 1 begin  
0007 1  
0008 1  
0009 1 *****  
0010 1 *  
0011 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0012 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0013 1 *  ALL RIGHTS RESERVED.  
0014 1 *  
0015 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0016 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0017 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0018 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0019 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0020 1 *  TRANSFERRED.  
0021 1 *  
0022 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0023 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0024 1 *  CORPORATION.  
0025 1 *  
0026 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0027 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0028 1 *  
0029 1 *****  
0030 1  
0031 1  
0032 1 ++  
0033 1 FACILITY:      System Management Utility Program  
0034 1  
0035 1 ABSTRACT:  
0036 1  
0037 1     This program allows the system manager to maintain the user  
0038 1     authorization file which contains usernames, passwords, quotas,  
0039 1     and defaults. The following functions are provided:  
0040 1  
0041 1     ADD - add a new user record to the authorization file (UAF)  
0042 1     COPY - copy a user record, give copied record a new name  
0043 1     DEFAULT - change a default value  
0044 1     EXIT - exit program and update file  
0045 1     HELP - ask for explanation  
0046 1     LIST - complete list of records to file  
0047 1     MODIFY - change one or more values for a user  
0048 1     REMOVE - remove a user record  
0049 1     RENAME - rename a user record (COPY; REMOVE)  
0050 1     SHOW - display the values from a user record  
0051 1  
0052 1 ENVIRONMENT:  
0053 1  
0054 1 AUTHOR:      Henry M. Levy, CREATION DATE: 1-June-1977  
0055 1  
0056 1 MODIFIED BY:  
0057 1
```



58	0058	1	V03-024	JRL0036	John R. Lawson, Jr.	07-Aug-1984 17:08
59	0059	1			Hide UPGRADE, DOWNGRADE, TMPJNL, PRMJNL privileges. This	
60	0060	1			is a temporary work-around; it is marked in the right-hand	
61	0061	1			margin with !** in routine PRINT_PRIV.	
62	0062	1				
63	0063	1	V03-023	JRL0027	John R. Lawson, Jr.	25-Jul-1984 11:01
64	0064	1			Add MODIFY/SYSTEM_PASSWORD=xxxxx to modify the system	
65	0065	1			password.	
66	0066	1				
67	0067	1	V03-022	JRL0029	John R. Lawson, Jr.	25-Jul-1984 10:47
68	0068	1			Find better names for UAF\$_NEWMSG10 and UAF\$_NEWMSG15	
69	0069	1				
70	0070	1	V03-021	JRL0020	John R. Lawson, Jr.	09-Jul-1984 15:51
71	0071	1			Bark when a proxy name is changed with the RENAME command.	
72	0072	1				
73	0073	1	V03-020	JRL0017	John R. Lawson, Jr.	02-Jul-1984 22:13
74	0074	1			Modify GET_UAF_RECORD so that it does not get the system	
75	0075	1			password record.	
76	0076	1				
77	0077	1	V03-029	JRL0013	John R. Lawson, Jr.	02-Jul-1984 12:28
78	0078	1			Display privileges of users in groups less than .EXESGL_SYSUIC	
79	0079	1			as 'ALL'.	
80	0080	1				
81	0081	1	V03-028	JRL0010	John R. Lawson, Jr.	25-Jun-1984 15:56
82	0082	1			Changed 'X'/'-' to '#'/'-' in primary/secondary access	
83	0083	1			display.	
84	0084	1				
85	0085	1	V03-027	JRL0008	John R. Lawson, Jr.	21-Jun-1984 14:00
86	0086	1			Add support for the /PWDEXPIRED qualifier (pre-expired	
87	0087	1			password).	
88	0088	1				
89	0089	1	V03-026	JRL0006	John R. Lawson, Jr.	20-Jun-1984 12:28
90	0090	1			Obliterate operator's console messages	
91	0091	1				
92	0092	1	V03-025	JRL0002	John R. Lawson, Jr.	15-Jun-1984 09:55
93	0093	1			Change all internal messages to calls to LIB\$SIGNAL and	
94	0094	1			the message utility -- place all messages in UAFMSG.MSG	
95	0095	1				
96	0096	1	V03-024	LY0494	Larry Yetto	11-JUN-1984 12:59
97	0097	1			Fix noise error message coming from ADD/ID/USER=* caused	
98	0098	1			by a flag not properly being set	
99	0099	1				
100	0100	1	V03-023	MHB0150	Mark Bramhall	2-May-1984
101	0101	1			Remove unused reference to TPARSE definitions.	
102	0102	1			Add DISPECONNECT flag.	
103	0103	1			Add security auditing for SYSUAF/NETUAF changes.	
104	0104	1				
105	0105	1	V03-022	LY0474	Larry Yetto	9-APR-1984 08:32
106	0106	1			Zero the login failure and last login fields	
107	0107	1			on a copy operation.	
108	0108	1				
109	0109	1	V03-021	LY0466	Larry Yetto	22-MAR-1984 13:52
110	0110	1			Add support for the rights data base functions	
111	0111	1				
112	0112	1	V03-020	ACG0397	Andrew C. Goldstein,	24-Feb-1984 23:21
113	0113	1			Clean up display formatting	
114	0114	1				



115 0115 1  
116 0116 1  
117 0117 1  
118 0118 1  
119 0119 1  
120 0120 1  
121 0121 1  
122 0122 1  
123 0123 1  
124 0124 1  
125 0125 1  
126 0126 1  
127 0127 1  
128 0128 1  
129 0129 1  
130 0130 1  
131 0131 1  
132 0132 1  
133 0133 1  
134 0134 1  
135 0135 1  
136 0136 1  
137 0137 1  
138 0138 1  
139 0139 1  
140 0140 1  
141 0141 1  
142 0142 1  
143 0143 1  
144 0144 1  
145 0145 1  
146 0146 1  
147 0147 1  
148 0148 1  
149 0149 1  
150 0150 1  
151 0151 1  
152 0152 1  
153 0153 1  
154 0154 1  
155 0155 1  
156 0156 1  
157 0157 1  
158 0158 1  
159 0159 1  
160 0160 1  
161 0161 1  
162 0162 1  
163 0163 1  
164 0164 1  
165 0165 1  
166 0166 1  
167 0167 1  
168 0168 1  
169 0169 1  
170 0170 1  
171 0171 1

V03-019 ACG0397 Andrew C. Goldstein, 6-Feb-1984 16:27  
Add DISREPORT to flags, clean up record locking

V03-018 ACG0388 Andrew C. Goldstein, 12-Jan-1984 19:21  
Add command input to handle new UAF features;  
general code cleanup

V03-017 ACG0385 Andrew C. Goldstein, 6-Jan-1984 18:28  
V4 UAF format change; remove read-only under installed  
SYSPRV feature; misc. code cleanups

V03-016 TMK0001 Todd M. Katz 10-Oct-1983  
Add JTQUOTA (job-wide logical name table creation quota)  
qualifier.

V03-015 LMP0153 L. Mark Pilant, 13-Sep-1983 11:57  
Add minimal support for alphanumeric UICs.

014 JWT0105 Jim Teague 30-Mar-1983  
Small changes to CLITABLES implementation.

013 JWT0104 Jim Teague 29-Mar-1983  
Add CLITABLES qualifier.

012 WMC0001 Wayne Cardoza 15-Mar-1983  
Add MAXDETACH qualifier.

011 JWT0097 Jim Teague 23-Feb-1983  
Fix RENAME problem with proxy entries.

010 JWT0096 Jim Teague 08-Feb-1983  
Log NETUAF changes to console, too.

009 JWT0087 Jim Teague 11-Jan-1983  
Change SYSWSQUOTA for created UAFs to 350

008 JWT0082 Jim Teague 05-Jan-1983  
Fix problem with LIST/PROXY.

007 JWT0079 Jim Teague 15-Dec-1982  
Enlarge output field for BYTLM; reset pending  
mail count for COPY operations.

006 JWT0072 Jim Teague 03-Dec-1982  
Add global longword which can be patched to  
enable/disable console logging of SYSUAF mods.

005 JWT0069 Jim Teague 24-Nov-1982  
Allow redefinition of sys\$output.

004 JWT0057 Jim Teague 21-Sep-1982  
Add a message to tell whether or not NETUAF was  
modified.

003 JWT0042 Jim Teague 15-Jul-1982  
Make SYSUAF.LIS and NETUAF.LIS world noread.

UAFMAIN  
V04-000

K 10  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 4  
(1)

: 172 0172 1 :  
: 173 0173 1 :  
: 174 0174 1 :  
: 175 0175 1 :  
: 176 0176 1 :  
: 177 0177 1 :  
: 178 0178 1 :  
: 179 0179 1 :  
: 180 0180 1 :--

002 JWT0036 Jim Teague 08-Jun-1982  
Add full wildcarding to show/proxy  
  
001 JWT0022 Jim Teague 17-Mar-1982  
Fix bug that caused failure to reparse command line for  
wildcard modifications. List default device on its own  
line for show/full.

```
182 0181 1 |
183 0182 1 | Require files
184 0183 1 |
185 0184 1 | require
186 0185 1 | 'lib$:uafreq';
187 0281 1 |
188 0282 1 |
189 0283 1 | INCLUDE FILES:
190 0284 1 |
191 0285 1 | library 'SYS$LIBRARY:LIB.L32';
192 0286 1 |
193 0287 1 |
194 0288 1 | TABLE OF CONTENTS:
195 0289 1 |
196 0290 1 |
197 0291 1 | forward routine
198 0292 1 | start,
199 0293 1 | setup : novalue,
200 0294 1 | add_uaf : novalue,
201 0295 1 | add_proxy : novalue,
202 0296 1 | remote_parse,
203 0297 1 | copy_uaf,
204 0298 1 | create_proxy : novalue,
205 0299 1 | modify_uaf : novalue,
206 0300 1 | modify_rec,
207 0301 1 | remove_uaf : novalue,
208 0302 1 | remove_proxy : novalue,
209 0303 1 | rename_uaf : novalue,
210 0304 1 | adjust_proxy : novalue,
211 0305 1 | default_uaf : novalue,
212 0306 1 | list_proxy : novalue,
213 0307 1 | list_uaf : novalue,
214 0308 1 | show_user_uaf : novalue,
215 0309 1 | show_proxy : novalue,
216 0310 1 | locate_proxy,
217 0311 1 | get_proxy_record,
218 0312 1 | display_proxy : novalue,
219 0313 1 | wild_user,
220 0314 1 | display_brief,
221 0315 1 | classify_priv,
222 0316 1 | display_full,
223 0317 1 | display_hours : novalue,
224 0318 1 | convert_time : novalue,
225 0319 1 | print_priv : novalue,
226 0320 1 | build_ini_recs : novalue,
227 0321 1 |
228 0322 1 | get_user_record,
229 0323 1 | locate_user,
230 0324 1 | get_uaf_record,
231 0325 1 | get_cmd_line,
232 0326 1 | ask : novalue,
233 0327 1 | fmt_sys_msg : novalue,
234 0328 1 | faout,
235 0329 1 | help_uaf : novalue,
236 0330 1 | exit_uaf : novalue,
237 0331 1 | SIGNAL SYNTAX : novalue,
238 0332 1 | acc$exit : novalue,
```

```
| controlling code
| open initial files
| insert new user record
| insert new proxy record
| parses "node: : remoteuser"
| copy user record
| create NETUAF.DAT proxy file
| update user record(s)
| update a user record action routine
| remove username from file
| remove a proxy record
| rename user record
| implicitly remove/update proxy record
| change default record
| list proxy entries in NETUAF.LIS
| list file routine
| display user record
| display proxy record at terminal
| access given proxy record(s)
| read single proxy record
| format and output a proxy entry
| user wild card routine
| writes a brief user display
| classifies contents of priv vector
| writes the full user display
| display hourly restrictions
| convert time value to string
| print privilege bits
| build initial file records for default
| and system manager
| get username and lookup record
| lookup user record in UAF
| routine to deal with record locking
| input user command line
| prompt terminal for input
| output system message file message
| output formatted message
| help routine
| normal exit routine
| missing qualifier
| exit and cleanup routine
```



```
: 239      0333 1      uaf$mod_sys_pwd : novalue,      ! modify the system password
: 240      0334 1      security_audit : novalue;      ! Perform a security audit
: 241      0335 1
: 242      0336 1 linkage
: 243      0337 1      fmg_match = jsb (register = 2, register = 3, register = 4,
: 244      0338 1      register = 5) : notused (10, 11);
: 245      0339 1
: 246      0340 1
: 247      0341 1      ! EXTERNAL REFERENCES:
: 248      0342 1
: 249      0343 1
: 250      0344 1 external literal
: 251      0345 1      cli$bufovf,
: 252      0346 1      cli$noclint;
: 253      0347 1
: 254      0348 1
: 255      0349 1 external routine
: 256      0350 1      lbr$output_help,
: 257      0351 1      lib$get_foreign,
: 258      0352 1      lib$get_input,
: 259      0353 1      lib$put_output,
: 260      0354 1      fmg$match_name : fmg_match,
: 261      0355 1      cli$dcl_parse,
: 262      0356 1      cli$dispatch,
: 263      0357 1      cli$present,
: 264      0358 1      cli$get_value,
: 265      0359 1      update_record,      ! modify all specified fields
: 266      0360 1      parse_wild,      ! parses a wildcarded user specification
: 267      0361 1      lgishpwd,      ! hash password routine
: 268      0362 1      uaf$add_ident_recbuf,
: 269      0363 1      uaf$build_holder,
: 270      0364 1      uaf$find_uic,
: 271      0365 1      uaf$remove_ident_recbuf,
: 272      0366 1      uaf$write_rights;
: 273      0367 1
: 274      0368 1 external
: 275      0369 1      EXE$GL_SYSUIC : long ,
: 276      0370 1      rdb_header_flag : byte ,
: 277      0371 1      rdb_list_flag : byte ,
: 278      0372 1      attributes : long ,
: 279      0373 1      holder : $bblock[8] ,
: 280      0374 1      ident : $bblock[4] ,
: 281      0375 1      authorize_commands,      ! AUTHORIZE command parse tables
: 282      0376 1      prv$ab_names;      ! address of privilege name table
: 283      0377 1
: 284      0378 1
: 285      0379 1
: 286      0380 1      ! MACROS:
: 287      0381 1
: 288      0382 1      macro
: 289      0383 1
: 290      M 0384 1      namelen (x, y) =
: 291      M 0385 1      begin
: 292      M 0386 1      builtin
: 293      M 0387 1      locc;
: 294      M 0388 1      register
: 295      M 0389 1      r0 = 0;
```

```
296 M 0390 1      locc (%ref(' '), %ref(x), y; r0);
297 M 0391 1      x = .r0
298      0392 1      end%,
299      0393 1
300 M 0394 1      cstring[] = (uplit byte (%charcount (%string (%remaining)),
301      0395 1      %string (%remaining)))%,
302      0396 1
303      0397 1      fatal[] = (fmt_sys_msg (%remaining); acc$exit ())%,
304      0398 1
305      0399 1      Macros to check for success or failure from RMS
306      0400 1
307      0401 1      rmsbad (string) = (not (rmserr = string)) %,
308      0402 1      rmsok (string) = (rmserr = string) %,
309      0403 1
310      0404 1      Macros to set up and write an FAO string.
311      0405 1
312 M 0406 1      faomac (faomsg)[] =
313 M 0407 1      begin
314 M 0408 1      faodsc[dsc$w_length] = . (faomsg)<0,8>;
315 M 0409 1      faodsc[dsc$a_pointer] = (faomsg) + 1;
316 M 0410 1      $fao (faodsc, rabptr[rab$w_rsz], disdsc $comma (%remaining)
317 M 0411 1      %remaining);
318 M 0412 1      $put (rab = .rabptr);
319      0413 1      end %,
320      0414 1
321      0415 1      $comma[] = , %,
322      0416 1
323 M 0417 1      output_null =
324 M 0418 1      begin
325 M 0419 1      rabptr[rab$w_rsz] = 0;
326 M 0420 1      $put (rab = .rabptr);
327      0421 1      end %,
328      0422 1
329      0423 1
330      0424 1      Macro to create string descriptor for command parameters and
331      0425 1      qualifiers
332      0426 1
333 M 0427 1      sd[a] =
334      0428 1      bind %name ('SD_',a) = $descriptor (a)%;
335      0429 1
336 P 0430 1      sd (
337 P 0431 1      'token1',      'token2',      'brief',      'full',
338 P 0432 1      'add_identifier',      'remove_identifier',
339 P 0433 1      'modify_identifier'
340      0434 1      );
341      0435 1
342      0436 1      field
343      0437 1      descr_fields =      ! Define the fields for a DESCRIPTOR
344      0438 1      set
345      0439 1      length = [dsc$w_length],
346      0440 1      dtype = [dsc$b_dtype],
347      0441 1      class = [dsc$b_class],
348      0442 1      pointer = [dsc$a_pointer]
349      0443 1      tes;
350      0444 1
351      0445 1      macro
352 M 0446 1      statdesc =
```



```
353 M 0447 1 $bblock [dsc$k_s_bln] field (descr_fields)
354 M 0448 1 preset ( [length] = 0,
355 M 0449 1 [dtype] = dsc$k_dtype_t,
356 M 0450 1 [class] = dsc$k_class_s,
357 M 0451 1 [pointer] = 0);
358 M 0452 1
359 M 0453 1 macro
360 M 0454 1 qualstr_desc (string) =
361 M 0455 1 $bblock [dsc$k_s_bln] field (descr_fields)
362 M 0456 1 preset ( [length] = (%charcount(string)),
363 M 0457 1 [dtype] = dsc$k_dtype_t,
364 M 0458 1 [class] = dsc$k_class_s,
365 M 0459 1 [pointer] = (uplit byte (%string(string)))));
366 M 0460 1 own
367 M 0461 1 sd_attrresource : qualstr_desc ('attributes.resource') ;
368 M 0462 1
369 M 0463 1
370 M 0464 1 EQUATED SYMBOLS:
371 M 0465 1
372 M 0466 1 literal
373 M 0467 1 cmdbufmax = 508, ! maximum command length
374 M 0468 1 false = 0, ! logical false
375 M 0469 1 true = 1, ! logical true
376 M 0470 1 update_records = 0, ! flag for proxy file adjustment
377 M 0471 1 remove_records = 1,
378 M 0472 1 copy_flag = 1, ! used in routine copy_uaf
379 M 0473 1 rename_flag = 1, ! used in routine rename_uaf
380 M 0474 1 byte_length = 8, ! bits per byte
381 M 0475 1 word_length = 16, ! bits per word
382 M 0476 1 long_length = 32, ! bits per longword
383 M 0477 1 retry_rlk = 8, ! number of retries for a locked record
384 M 0478 1 sleep_rlk = 500, ! ms to sleep before retrying
385 M 0479 1 cr = 13,
386 M 0480 1 lf = 10,
387 M 0481 1 zero = 0,
388 M 0482 1 cmdbuflen = 1024; ! size of user command buffer
389 M 0483 1
390 M 0484 1 global literal
391 M 0485 1 encrypt = uaf$c_purdy_v, ! encryption algorithm to use
392 M 0486 1 disbuflen = 132; ! size of display file output buffer
393 M 0487 1
394 M 0488 1 bind
395 M 0489 1 sysuaf_string = uplit byte ('SYSUAF'),
396 M 0490 1 netuaf_string = uplit byte ('NETUAF'),
397 M 0491 1 mod_act_dsc = $descriptor ('modified'),
398 M 0492 1 add_act_dsc = $descriptor ('added'),
399 M 0493 1 rem_act_dsc = $descriptor ('removed'),
400 M 0494 1 fao_lin_dsc = $descriptor ('PID= !XL !AS !AS !AS record !AS on !XD'),
401 M 0495 1 dbl_colon = $descriptor ('::'),
402 M 0496 1
403 M 0497 1 Define the system delta time to sleep before retrying to GET a locked record.
404 M 0498 1
405 M 0499 1 wakedelta = uplit long (-10*1000*sleep_rlk,-1),
406 M 0500 1
407 M 0501 1 Default values for authorization file record. These values are
408 M 0502 1 only used when a new authorization file is created. If the file
409 M 0503 1 already exists, the default values are read from the first file
```



```
410 0504 1 record.
411 0505 1
412 0506 1 defuser = cstring ('DEFAULT'), ! default username
413 0507 1 defpass = cstring ('USER'), ! default password
414 0508 1 defclitabl = cstring ('DCLTABLES'), ! default CLI tables
415 0509 1 defact = cstring (''), ! default account name
416 0510 1 defcli = cstring ('DCL'), ! default command interpreter
417 0511 1 defowner = cstring (''), ! owner's name
418 0512 1 deflgicmd = cstring (''), ! default login command file
419 0513 1 defgrp = %'200', ! default group
420 0514 1 defmem = %'200', ! default member
421 0515 1 defdir = cstring ('[USER]'), ! default directory name
422 0516 1 defdev = cstring (''), ! default device name
423 0517 1 defbiolm = 6, ! default buffered I/O limit
424 0518 1 defbytlm = 4096, ! default buffered I/O buffer space
425 0519 1 defdiolm = 6, ! default direct I/O limit
426 0520 1 deffillm = 20, ! default open file limit
427 0521 1 defflags = 0, ! default flag bits
428 0522 1 deftcnt = 10, ! default time queue entries
429 0523 1 defprcct = 2, ! default subprocess count
430 0524 1 defpri = 4, ! default process priority
431 0525 1 defquepri = 4, ! default queue priority
432 0526 1 defwsquota = 200, ! default working set limit
433 0527 1 defdfwscnt = 150, ! "default" working set default size
434 0528 1 defwsextent = 500, ! default working set extent
435 0529 1 defcputim = 0, ! default CPU time quota
436 0530 1 defastlm = 10, ! default AST queue limit
437 0531 1 defpgflquota = 10000, ! default paging file limit in pages
438 0532 1 defenqlm = 10, ! default enqueue limit
439 0533 1 defpbytlm = 0, ! default paged buffer I/O limit
440 0534 1 defshrfillm = 0, ! default shared file limit
441 0535 1 defmaxjobs = 0, ! default maximum concurrent jobs
442 0536 1 defmaxacctjobs = 0, ! default maximum concurrent group jobs
443 0537 1 defmaxdetach = 0, ! default maximum detached processes
444 0538 1 defjtquota = 1024, ! default job-wide logical table quota
445 0539 2 defprimedays = ( ! Sat, Sun are default non-prime days
446 0540 2 (1 ^ $bitposition (uaf$v_saturday)) or
447 0541 3 (1 ^ $bitposition (uaf$v_sunday))
448 0542 1 ),
449 0543 1 defhours = 0, ! default all hours to allow access
450 0544 1 defpriv = uplit ( ! default privilege vector
451 0545 2 (
452 0546 2 (1 ^ $bitposition (prv$v_netmbx)) or
453 0547 3 (1 ^ $bitposition (prv$v_tmmbx))
454 0548 1 ), 0),
455 0549 1 defpwdlength = 6, ! default min password length
456 0550 1 defpwdlife = uplit (0,0), ! default password lifetime
457 0551 1
458 0552 1
459 0553 1 The following are the default values for the SYSTEM user. When
460 0554 1 a new file is created, a system manager record is created.
461 0555 1
462 0556 1 sysuser = cstring ('SYSTEM'),
463 0557 1 syspass = cstring ('MANAGER'),
464 0558 1 sysclitabl = cstring ('DCLTABLES'),
465 0559 1 sysact = cstring ('SYSTEM'),
466 0560 1 syscli = cstring ('DCL').
```

```
467 0561 1 sysowner = cstring ('SYSTEM MANAGER'),
468 0562 1 syslgicmd = cstring (''),
469 0563 1 sysgrp = %'1',
470 0564 1 sysmem = %'4',
471 0565 1 sysdir = cstring ('[SYSMGR]'),
472 0566 1 sysdev = cstring (''),
473 0567 1 sysbiolm = 12,
474 0568 1 sysbytlm = 20480,
475 0569 1 sysdiolm = 12,
476 0570 1 sysfillm = 20,
477 0571 1 sysflags = 0,
478 0572 1 systqcnt = 20,
479 0573 1 sysprcct = 10,
480 0574 1 syspri = 4,
481 0575 1 sysquepri = 4,
482 0576 1 syswsquota = 350,
483 0577 1 syswsextent = 1024,
484 0578 1 sysdfuscnt = 150,
485 0579 1 syscputim = 0,
486 0580 1 sysastlm = 20,
487 0581 1 syspgflquota = 10000,
488 0582 1 sysenqlm = 20,
489 0583 1 syspbytlm = 0,
490 0584 1 sysshrfillm = 0,
491 0585 1 sysmaxjobs = 0,
492 0586 1 sysmaxdetach = 0,
493 0587 1 sysjtquota = 1024,
494 0588 1 sysmaxacctjobs = 0,
495 0589 2 sysprimedays = ( ! Sat, Sun are default non-prime days
496 0590 2 (1 ^ $bitposition (uaf$v_saturday)) or
497 0591 2 (1 ^ $bitposition (uaf$v_sunday))
498 0592 1 ),
499 0593 1 syshours = 0,
500 0594 1 syspriv = uplit (rep 2 of (%'FFFFFFFF')),
501 0595 1 syspwdlength = 8, ! default min password length
502 0596 1 syspwdlife = uplit (0,0); ! default password lifetime
503 0597 1
504 0598 1
505 0599 1 ! GLOBAL STORAGE - must be before OWN for initialization purposes
506 0600 1
507 0601 1 global
508 0602 1 disbuf : vector [disbuflen, byte], ! display buffer
509 0603 1 disdsc : block [8, byte] initial (disbuflen, disbuf) ;
510 0604 1
511 0605 1
512 0606 1 ! OWN STORAGE:
513 0607 1
514 0608 1 own
515 0609 1 username_buf : block [uaf$s_username, byte],
516 0610 1 pcb_sts : bitvector [32],
517 0611 1 pid,
518 0612 1 username_dsc : vector [2] initial (0, username_buf),
519 0613 1 recname_dsc : vector [2],
520 0614 1 file_dsc : vector [2] initial (6),
521 0615 1 mod_default, ! DEFAULT record being modified by MODIFY command
522 0616 1 modify_flag : long, ! SYSUAF modified
523 0617 1 netuaf_modified, ! NETUAF modified
```

```
0618 1 rename_ph2 : byte initial (false),
0619 1 olduserlen : long,
0620 1 oldusername : vector [uaf$s_username,byte],
0621 1 newuserlen : long,
0622 1 newusername : vector [uaf$s_username,byte],
0623 1 cmdbuf : vector [cmdbuflen, byte], ! command buffer
0624 1 default_size : word,
0625 1 default_record : block [uaf$c_length, byte], ! default record held here
0626 1 pwddsc : block [8, byte], ! password descriptor
0627 1 insize : long, ! number of input chars
0628 1 brief_flag : long, ! display option
0629 1 full_flag : long, ! display option
0630 1 header_flag : long, ! output header or not?
0631 1
0632 1
0633 1 infab : $fab ( ! FAB for terminal I/O
0634 1 fac = (get, put),
0635 1 rat = cr,
0636 1 fnm = 'SYS$INPUT'
0637 1 ),
0638 1
0639 1 inrab : $rab ( ! RAB for terminal I/O
0640 1 rac = seq,
0641 1 rop = pmt,
0642 1 fab = infab
0643 1 ),
0644 1
0645 1 outfab : $fab (
0646 1 fac = (put),
0647 1 rat = cr,
0648 1 fnm = 'SYS$OUTPUT'
0649 1 ),
0650 1
0651 1 lstnam : $nam (, ! needed for the DLT option
0652 1
0653 1 lstpro : $xabpro (
0654 1 pro = (rwd,rwd,rw)
0655 1 ),
0656 1
0657 1 lstfab : $fab ( ! FAB for UAF Listing file
0658 1 fac = put,
0659 1 rat = cr,
0660 1 fnm = 'SYSUAF.LIS',
0661 1 shr = nil,
0662 1 org = seq,
0663 1 rfm = var,
0664 1 mrs = disbuflen,
0665 1 nam = lstnam,
0666 1 xab = lstpro
0667 1 ),
0668 1
0669 1 lstrab : $rab ( ! RAB for UAF Listing file
0670 1 rac = seq,
0671 1 rbf = disbuf,
0672 1 fab = lstfab
0673 1 ),
0674 1
```



```
581      0675 1
582      0676 1
583      P 0677 1
584      P 0678 1
585      0679 1
586      0680 1
587      P 0681 1
588      P 0682 1
589      P 0683 1
590      P 0684 1
591      P 0685 1
592      P 0686 1
593      P 0687 1
594      P 0688 1
595      P 0689 1
596      P 0690 1
597      0691 1
598      0692 1
599      P 0693 1
600      P 0694 1
601      P 0695 1
602      P 0696 1
603      0697 1
604      0698 1
605      P 0699 1
606      P 0700 1
607      P 0701 1
608      P 0702 1
609      P 0703 1
610      P 0704 1
611      0705 1
612      0706 1
613      P 0707 1
614      P 0708 1
615      P 0709 1
616      P 0710 1
617      P 0711 1
618      P 0712 1
619      P 0713 1
620      0714 1
621      0715 1
622      P 0716 1
623      P 0717 1
624      P 0718 1
625      P 0719 1
626      P 0720 1
627      0721 1
628      0722 1
629      P 0723 1
630      P 0724 1
631      P 0725 1
632      0726 1
633      0727 1
634      P 0728 1
635      P 0729 1
636      P 0730 1
637      P 0731 1
```

```
nlstnam : $nam (
nlstpro : $xabpro (
    pro = (rwd,rwd,rw)
),
nlstfab : $fab (
    ! FAB for NETUAF Listing file
    fac = put,
    rat = cr,
    fnm = 'NETUAF.LIS',
    shr = nil,
    org = seq,
    rfm = var,
    mrs = disbuflen,
    nam = nlstnam,
    xab = nlstpro
),
nlstrab : $rab (
    ! RAB for NETUAF Listing file
    rac = seq,
    rbf = disbuf,
    fab = nlstfab
),
uafkey2 : $xabkey (
    ! XAB for User number key
    ! alternate key
    kref = 2,
    pos0 = $byteoffset (uaf$u_mem),
    siz0 = 2,
    dtp = bn2,
    flg = (chg,dup)
),
uafkey1 : $xabkey (
    ! XAB for Group number key
    ! alternate key
    kref = 1,
    pos0 = $byteoffset (uaf$l_uic),
    siz0 = 4,
    dtp = bn4,
    flg = (chg,dup),
    nxt = uafkey2
),
uafkey0 : $xabkey (
    ! XAB for USERNAME key
    ! primary key
    kref = 0,
    pos0 = $byteoffset (uaf$t_username),
    siz0 = uaf$s_username,
    nxt = uafkeyT
),
uafpro : $xabpro (
    ! XAB for file protection
    ! deny world access
    pro = (rwd, rwd),
    nxt = uafkey0
),
uaffab : $fab (
    ! FAB for work file
    fop = cif,
    fac = (get, put, del, upd),
    fnm = 'SYSUAF',
```

```
638 P 0732 1 dnm = '.DAT',
639 P 0733 1 shr = (get, put, del, upd),
640 P 0734 1 org = idx,
641 P 0735 1 rfm = var,
642 P 0736 1 mrs = uaf$c_length,
643 P 0737 1 alq = 10,
644 P 0738 1 deq = 10,
645 P 0739 1 xab = uafpro
646 0740 1 ),
647 0741 1
648 0742 1 FAB for NETUAF Proxy Login File
649 0743 1
650 0744 1
651 P 0745 1 nafkey1 : $xabkey (
652 P 0746 1 kref = 1,
653 P 0747 1 pos0 = $byteoffset (naf$t_localuser),
654 P 0748 1 siz0 = naf$s_localuser,
655 P 0749 1 flg = (chg,dup)
656 0750 1 ),
657 0751 1
658 P 0752 1 nafkey0 : $xabkey (
659 P 0753 1 kref = 0,
660 P 0754 1 pos0 = $byteoffset (naf$t_remname),
661 P 0755 1 siz0 = naf$s_remname,
662 P 0756 1 nxt = nafkey1
663 0757 1 ),
664 0758 1
665 P 0759 1 nafpro : $xabpro (
666 P 0760 1 pro = (rwed, rwed),
667 P 0761 1 nxt = nafkey0
668 0762 1 ),
669 0763 1
670 P 0764 1 naffab : $fab ( ! FAB for NETUAF
671 P 0765 1 fop = cif,
672 P 0766 1 fac = (get, put, del, upd),
673 P 0767 1 fnm = 'NETUAF',
674 P 0768 1 dnm = '.DAT',
675 P 0769 1 shr = (get, put, del, upd),
676 P 0770 1 org = idx,
677 P 0771 1 rfm = fix,
678 P 0772 1 mrs = naff$c_length,
679 P 0773 1 alq = 10,
680 P 0774 1 deq = 10,
681 P 0775 1 xab = nafpro
682 0776 1 );
683 0777 1
684 0778 1
685 0779 1 GLOBAL STORAGE:
686 0780 1
687 0781 1 global
688 0782 1 faodsc : block [8, byte], ! gen'l purpose fao string desc
689 0783 1 rabptr : ref block [rab$c_bln, byte], ! RAB for output
690 0784 1 uaf$gq_sysuaff : block [nsa_s_sysuaff, byte], ! SYSUAF auditing flags
691 0785 1 uaf$gl_ctlmsk : block [2], ! Control mask for AUTHORIZE
692 0786 1 by account : long initial (false), ! processing by account
693 0787 1 match_token : vector [naf$s_remname+2, byte], ! Saved match token
694 0788 1 match_tokenlen : long,
```

```
695      0789 1      wild_netuser,      ! wildcard access to proxy entries
696      0790 1      call_count      : long initial (0),      ! counter for reprocessing cmd line
697      0791 1      tokendsc      : block [8, byte] preset ( [dsc$b_class]=dsc$b_class_d),
698      0792 1      cmdlindsc      : block [8, byte],
699      0793 1      netuaf_exists,      ! A self-explanatory flag...
700      0794 1      rdb_exists      : long,
701      0795 1      rmserr      : long,      ! save rms error codes here
702      0796 1      rightslist_modified : byte,      ! RIGHTSLIST modified flag
703      0797 1
704      0798 1      Record buffer for file I/O. Records are generally read into RECBUF,
705      0799 1      modified, and output.
706      0800 1
707      0801 1      recbuf      : block [uaf$c_length, byte],
708      0802 1      netbuf      : block [naf$c_length, byte],
709      0803 1
710      0804 1      UAFRAB is global to allow UPDATE_RECORD to modify RAB$W_RSZ.
711      0805 1
712      P 0806 1      uafrab : $rab (! RAB for work file
713      P 0807 1          krf = 0,
714      P 0808 1          kbf = recbuf [uaf$t_username],
715      P 0809 1          ksz = uaf$s_username,
716      P 0810 1          usz = uaf$c_length,
717      P 0811 1          fab = uaffab
718      0812 1      ),
719      0813 1
720      0814 1      RAB for NETUAF Proxy Login File
721      0815 1
722      P 0816 1      nafrab : $rab (
723      P 0817 1          krf = 0,
724      P 0818 1          kbf = netbuf [naf$t_remname],
725      P 0819 1          ksz = naf$s_remname,
726      P 0820 1          usz = naf$c_length,
727      P 0821 1          rsz = naf$c_length,
728      P 0822 1          rac = key,
729      P 0823 1          fab = naf$fab
730      0824 1      ),
731      0825 1
732      0826 1
733      0827 1      time_buf      : block [8,byte],      ! current system time
734      0828 1      pwd_flag      : long,      ! Password default flag
735      0829 1
736      P 0830 1      outrab : $rab (
737      P 0831 1          rac = seq,
738      P 0832 1          rbf = disbuf,
739      P 0833 1          fab = outfab
740      0834 1      ),
741      0835 1
742      0836 1
743      0837 1      Flag signaling to WILD_USER that a match was found. This flag
744      0838 1      is set by the action routine called by WILD_USER.
745      0839 1
746      0840 1      found_match,      ! found at least one wildcard match
747      0841 1
748      0842 1
749      0843 1      Wildcard parsing flags set by PARSE_WILD for use in WILD_USER.
750      0844 1
751      0845 1      uic_flag      : long,
```



```
0846 1      grp_wild      : long,
0847 1      mem_wild      : long,
0848 1      str_wild      : long;
0849 1
0850 1 global bind
0851 1      tokenlen      = tokendsc [dsc$w_length] : word,
0852 1      tokenptr      = tokendsc [dsc$a_pointer],
0853 1      rec_user_dsc  = uplit (uaf$s_username, recbuf [uaf$t_username]),
0854 1      rec_encrypt_dsc = uplit (uaf$s_pwd, recbuf [uaf$q_pwd]),
0855 1      symbol_str    = cstring ('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$');
0856 1                      ! valid username characters
0857 1
0858 1      !
0859 1      Prompt strings
0860 1      !
0861 1
0862 1 bind
0863 1
0864 1      accprmt      = cstring (%char (lf), 'UAF> '),
0865 1      accprmt2     = cstring (%char (lf), '- '),
0866 1      newmsg20     = cstring ('Do you want to create a new file? ');
0867 1
0868 1      !
0869 1      External messages
0870 1      !
0871 1
0872 1 external routine
0873 1
0874 1      LIB$SIGNAL;
0875 1
0876 1 external literal
0877 1
0878 1      UAF$_ADDERR,      UAF$_ADDMSG,      UAF$_BADNODFORM,
0879 1      UAF$_BADSPC,      UAF$_BADUSR,      UAF$_CLIWARNMSG,
0880 1      UAF$_CMDTOOLONG,  UAF$_CONERR,      UAF$_COPMSG,
0881 1      UAF$_CREERR,      UAF$_DEFERR,      UAF$_DEFPWD,
0882 1      UAF$_DONEMSG,     UAF$_GETERR,      UAF$_HELPERR,
0883 1      UAF$_INVCMD,      UAF$_INVRSP,      UAF$_INVUSERNAME,
0884 1      UAF$_KEYNOTFND,   UAF$_KEYNOTUNQ,   UAF$_LSTERR,
0885 1      UAF$_LSTMSG1,     UAF$_LSTMSG2,     UAF$_MDFYERR,
0886 1      UAF$_MDFYMSG,     UAF$_NAFADDERR,   UAF$_NAFADDMSG,
0887 1      UAF$_NAFAEX,      UAF$_NAFCONERR,   UAF$_NAFCREERR,
0888 1      UAF$_NAFDNE,      UAF$_NAFDONEMSG,  UAF$_NAFNOMODS,
0889 1      UAF$_NAFUAEERR,   UAF$_NAMETOOBIG,  UAF$_NETLSTMSG,
0890 1      UAF$_NAOIFL,      UAF$_NAONAF,      UAF$_NOARG,
0891 1      UAF$_NODEFPWD,    UAF$_NODTOOBIG,   UAF$_NOMODS,
0892 1      UAF$_NOTUNQ,      UAF$_NOUSERNAME,  UAF$_PREMSG,
0893 1      UAF$_PUTERR,      UAF$_RDBDONEMSG,  UAF$_RDBMDFYERR,
0894 1      UAF$_RDBMDFYERRU, UAF$_RDBMDFYMSG,  UAF$_RDBNOMODS,
0895 1      UAF$_REMDEF,      UAF$_REHERR,      UAF$_REMMSG,
0896 1      UAF$_REMSYS,      UAF$_RENDEF,      UAF$_RENMSG,
0897 1      UAF$_RENSYS,      UAF$_RONLY,       UAF$_SHOWERR,
0898 1      UAF$_SYSMSG1,     UAF$_SYSMSG2,     UAF$_UAEERR,
0899 1      UAF$_UICERR,      UAF$_ZISQUAL,     UAF$_ZZPRACREN;
0900 1
```

start - controlling code

```

808 0901 1 %sbttl 'start - controlling code'
809 0902 1 routine start =
810 0903 begin
811 0904
812 0905 ++
813 0906
814 0907 FUNCTIONAL DESCRIPTION:
815 0908
816 0909 Main procedure of AUTHORIZE. Call SETUP to initialize
817 0910 all needed files. Prompt the user for the functions which
818 0911 he/she wants, and call the proper function service routine.
819 0912
820 0913 INPUTS:
821 0914
822 0915 none
823 0916
824 0917 IMPLICIT INPUTS:
825 0918
826 0919 none
827 0920
828 0921 OUTPUTS:
829 0922
830 0923 None
831 0924
832 0925 IMPLICIT OUTPUTS:
833 0926
834 0927 none
835 0928
836 0929 ROUTINE VALUE:
837 0930
838 0931 none
839 0932
840 0933 SIDE EFFECTS:
841 0934
842 0935 none
843 0936 --
844 0937
845 0938 own
846 0939 status;
847 0940
848 0941 map
849 0942 cmdlindsc : vector;
850 0943
851 0944 bind
852 0945 foreign_cmdlindsc = uplit (cmdbuflen, cmdbuf);
853 0946
854 0947 rightslist_modified = false;
855 0948 netuaf_modified = false;
856 0949 modify_flag = false;
857 0950
858 0951 !
859 0952 Set up terminal I/O
860 0953
861 0954
862 0955 if rmsbad (status = $open (fab = infab))
863 0956 then
864 0957 signal_stop (.status);
```

! note no modifications

start - controlling code

```

865 0958
866 0959 if rmsbad (status = $connect (rab = inrab))
867 0960 then
868 0961     signal_stop (.status);
869 0962
870 0963 $create (fab = outfab);
871 0964 $connect (rab = outrab);
872 0965
873 0966 setup ();
874 0967
875 0968
876 0969
877 0970
878 0971
879 0972
880 0973 if lib$get_foreign (foreign_cmdlindsc, 0, cmdlindsc) and .cmdlindsc [0] neq 0
881 0974 then
882 0975
883 0976     If defined foreign, and there are commands on the line...
884 0977
885 0978     begin
886 0979         cmdlindsc [1] = cmdbuf;
887 0980
888 0981         if (status = cli$dcl_parse (cmdlindsc, authorize_commands))
889 0982         then
890 0983             begin
891 0984                 cli$dispatch ();
892 0985                 return true;
893 0986             end
894 0987         else
895 0988
896 0989             See if no CLINT exists (Kludge City, USA 37916)
897 0990
898 0991             if .status eql cli$_noclint
899 0992             then
900 0993                 signal_stop (cli$_noclint)
901 0994             else
902 0995                 acc$exit ()
903 0996             end;
904 0997
905 0998 while true
906 0999 do
907 1000     begin
908 1001
909 1002
910 1003         Input the command line, taking care of continuations. Pull off
911 1004         the first token, assuming it is the command name, and look it up
912 1005         in the table of commands.
913 1006
914 1007
915 1008
916 1009
917 1010         if get_cmd_line ()
918 1011         then
919 1012             if (status = cli$dcl_parse (cmdlindsc, authorize_commands))
920 1013             then
921 1014                 begin
922                     ch$fill (0, uaf$s_flags, uaf$gl_ctlmsk);
923                     cli$dispatch ();

```



UAFMAIN  
V04-000

start - controlling code

L 11  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 18  
(3)

```

: 922      1015 4      end
: 923      1016      else
: 924      1017          if .status eql cli$_noclint
: 925      1018          then
: 926      1019              signal_stop (cli$_noclint)
: 927      1020          end;
: 928      1021      return true;
: 929      1022
: 930      1023
: 931      1024 1 end;

```

```

                                .TITLE  UAFMAIN
                                .IDENT   \V04-000\
                                .PSECT   $SPLITS,NOWRT,NOEXE,2
                                :
31 6E 65 6B 6F 74 00000 P.AAB: .ASCII  \token1\
                                :
                                00006 .BLKB  2
                                00008 P.AAA: .LONG  6
                                0000C .ADDRESS P.AAB
32 6E 65 6B 6F 74 00010 P.AAD: .ASCII  \token2\
                                :
                                00016 .BLKB  2
                                00018 P.AAC: .LONG  6
                                0001C .ADDRESS P.AAD
66 65 69 72 62 00020 P.AAF: .ASCII  \brief\
                                :
                                00025 .BLKB  3
                                00028 P.AAE: .LONG  5
                                0002C .ADDRESS P.AAF
6C 6C 75 66 00030 P.AAH: .ASCII  \full\
                                :
                                00034 P.AAG: .LONG  4
                                00038 .ADDRESS P.AAH
72 65 69 66 69 74 6E 65 64 69 5F 64 64 61 0003C P.AAJ: .ASCII  \add_identifier\
                                :
                                0004A .BLKB  2
                                0004C P.AAI: .LONG  14
                                00050 .ADDRESS P.AAJ
69 66 69 74 6E 65 64 69 5F 65 76 6F 6D 65 72 00054 P.AAL: .ASCII  \remove_identifier\
                                :
                                72 65 00063 .BLKB  3
                                00065 .LONG  17
                                00068 P.AAK: .ADDRESS P.AAL
69 66 69 74 6E 65 64 69 5F 79 66 69 64 6F 6D 00070 P.AAN: .ASCII  \modify_identifier\
                                :
                                72 65 0007F .BLKB  3
                                00081 .LONG  17
                                00084 P.AAM: .ADDRESS P.AAN
6F 73 65 72 2E 73 65 74 75 62 69 72 74 74 61 0008C P.AAO: .ASCII  \attributes.resource\
                                :
                                65 63 72 75 0009B .BLKB  1
                                46 41 55 53 59 53 0009F P.AAP: .ASCII  \SYSUAF\
                                46 41 55 54 45 4E 000A5 P.AAQ: .ASCII  \NETUAF\
                                64 65 69 66 69 64 6F 6D 000AB P.AAS: .ASCII  \modified\
                                :
                                000B3 .BLKB  8
                                000B4 P.AAR: .LONG  8
                                000B8 .ADDRESS P.AAS
64 65 64 64 61 000BC P.AAU: .ASCII  \added\
                                :
                                000C1 .BLKB  3
                                000C4 P.AAT: .LONG  5
                                :

```

Address	Disassembly	Comment
00000000	000C8	.ADDRESS P.AAU
00000001	000CC	.ASCII \removed\
00000002	000D3	.BLKB 1
00000003	000D4	.LONG 7
00000004	000D8	.ADDRESS P.AAW
00000005	000DC	.ASCII \PID=!XL !AS !AS !AS record !AS on !%D\
00000006	000EB	
00000007	000FA	
00000008	00101	.BLKB 3
00000009	00104	.LONG 37
0000000A	00108	.ADDRESS P.AAY
0000000B	0010C	.ASCII \::\
0000000C	0010E	.BLKB 2
0000000D	00110	.LONG 2
0000000E	00114	.ADDRESS P.ABA
0000000F	00118	.LONG -5000000, -1
00000010	00120	.BYTE 7
00000011	00121	.ASCII \DEFAULT\
00000012	00128	.BYTE 4
00000013	00129	.ASCII \USER\
00000014	0012D	.BYTE 9
00000015	0012E	.ASCII \DCLTABLES\
00000016	00137	.BYTE 0
00000017	00138	.BLKB 0
00000018	00138	.BYTE 3
00000019	00139	.ASCII \DCL\
0000001A	0013C	.BYTE 0
0000001B	0013D	.BLKB 0
0000001C	0013D	.BYTE 0
0000001D	0013E	.BLKB 0
0000001E	0013E	.BYTE 6
0000001F	0013F	.ASCII \[USER]\
00000020	00145	.BYTE 0
00000021	00146	.BLKB 0
00000022	00146	.BLKB 2
00000023	00148	.LONG 1081344, 0
00000024	00150	.LONG 0, 0
00000025	00158	.BYTE 6
00000026	00159	.ASCII \SYSTEM\
00000027	0015F	.BYTE 7
00000028	00160	.ASCII \MANAGER\
00000029	00167	.BYTE 9
0000002A	00168	.ASCII \DCLTABLES\
0000002B	00171	.BYTE 6
0000002C	00172	.ASCII \SYSTEM\
0000002D	00178	.BYTE 3
0000002E	00179	.ASCII \DCL\
0000002F	0017C	.BYTE 14
00000030	0017D	.ASCII \SYSTEM MANAGER\
00000031	00188	.BYTE 0
00000032	0018C	.BLKB 0
00000033	0018C	.BYTE 8
00000034	0018D	.ASCII \[SYSMGR]\
00000035	00195	.BYTE 0
00000036	00196	.BLKB 0
00000037	00196	.BLKB 2
00000038	00198	.LONG -1[2]

```

00000000 00000000 001A0 P.ABX: .LONG 0, 0
54 54 55 50 4E 49 24 53 59 53 001A8 P.ABY: .ASCII \SYSS$INPUT\
53 55 50 54 55 4F 24 53 59 53 001B1 P.ABZ: .ASCII \SYSS$OUTPUT\
53 49 4C 2E 46 41 55 53 59 53 001B8 P.ACA: .ASCII \SYSUAF.LIS\
53 49 4C 2E 46 41 55 53 45 4E 001C5 P.ACB: .ASCII \NETUAF.LIS\
46 41 55 53 59 53 001CF P.ACC: .ASCII \SYSUAF\
46 41 55 53 41 44 2E 001D5 P.ACD: .ASCII \.DAT\
46 41 55 53 54 45 4E 001D9 P.ACE: .ASCII \NETUAF\
54 41 44 2E 001DF P.ACF: .ASCII \.DAT\
001E3 .BLKB 1
00000020 001E4 P.ACG: .LONG 32
00000000 001E8 .ADDRESS RECBUF+4
00000008 001EC P.ACH: .LONG 8
00000000 001F0 .ADDRESS RECBUF+340
4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 001F4 P.ACI: .BYTE 38
33 32 31 30 5A 59 58 57 56 55 54 53 52 51 50 001F5 .ASCII \ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$_\
5F 24 39 38 37 36 35 34 00204
20 3E 46 41 55 0A 00213
20 5F 0A 00218 P.ACJ: .BYTE 6
0021C .ASCII <10>\UAF> \
00222 P.ACK: .BYTE 3
00223 .ASCII <10>\_ \
20 6F 74 20 74 6E 61 77 20 75 6F 79 20 6F 44 00226 P.ACL: .BYTE 34
69 66 20 77 65 6E 20 61 20 65 74 61 65 72 63 00227 .ASCII \Do you want to create a new file? \
20 3F 65 6C 00236
00245
00249 .BLKB 3
00000400 0024C P.ACM: .LONG 1024
00000000 00250 .ADDRESS CMDBUF
.PSECT $OWNS,NOEXE,2
0013 00000 SD_ATTRIBRESOURCE:
01 0E 00002 .WORD 19
00000000 00004 .BYTE 14, 1
00008 USERNAME_BUF:
00028 PCB_STS: .BLKB 4
0002C PID: .BLKB 4
00000000 00030 USERNAME_DSC:
00034 .LONG 0
00038 RECNAME_DSC:
00000006 00040 FILE_DSC:
00044 .LONG 6
00048 MOD_DEFAULT: .BLKB 4
0004C MODIFY_FLAG: .BLKB 4
00050 NETUAF_MODIFIED: .BLKB 4
00 00054 RENAME_PH2:
00055 .BYTE 0
.BKLB 3

```



	00058	OLDUSERLEN:		
		.BLKB	4	
	0005C	OLDUSERNAME:		
		.BLKB	32	
	0007C	NEWUSERLEN:		
		.BLKB	4	
	00080	NEWUSERNAME:		
		.BLKB	32	
	000A0	CMDBUF:	.BLKB	1024
	004A0	DEFAULT_SIZE:		
		.BLKB	2	
	004A2		.BLKB	2
	004A4	DEFAULT_RECORD:		
		.BLKB	1412	
	00A28	PWDDSC:	.BLKB	8
	00A30	INSIZE:	.BLKB	4
	00A34	BRIEF_FLAG:		
		.BLKB	4	
	00A38	FULL_FLAG:		
		.BLKB	4	
	00A3C	HEADER_FLAG:		
		.BLKB	4	
03	00A40	INFAB:	.BYTE	3
50	00A41		.BYTE	80
0000	00A42		.WORD	0
00000000	00A44		.LONG	0
00000000	00A48		.LONG	0
00000000	00A4C		.LONG	0
00000000	00A50		.LONG	0
0000	00A54		.WORD	0
03	00A56		.BYTE	3
00	00A57		.BYTE	0
00000000	00A58		.LONG	0
00	00A5C		.BYTE	0
00	00A5D		.BYTE	0
02	00A5E		.BYTE	2
02	00A5F		.BYTE	2
00000000	00A60		.LONG	0
00000000	00A64		.LONG	0
00000000	00A68		.LONG	0
00000000	00A6C		.ADDRESS	P.ABY
00000000	00A70		.LONG	0
09	00A74		.BYTE	9
00	00A75		.BYTE	0
0000	00A76		.WORD	0
00000000	00A78		.LONG	0
0000	00A7C		.WORD	0
00	00A7E		.BYTE	0
00	00A7F		.BYTE	0
00000000	00A80		.LONG	0
00000000	00A84		.LONG	0
0000	00A88		.WORD	0
00	00A8A		.BYTE	0
00	00A8B		.BYTE	0
00000000	00A8C		.LONG	0
01	00A90	INRAB:	.BYTE	1
44	00A91		.BYTE	68

0000	00A92		.WORD	0
40000000	00A94		.LONG	1073741824
00000000	00A98		.LONG	0
00000000	00A9C		.LONG	0
0000	00AA0		.WORD	0[3]
0000	00AA6		.WORD	0
00000000	00AA8		.LONG	0
0000	00AAC		.WORD	0
00	00AAE		.BYTE	0
00	00AAF		.BYTE	0
0000	00AB0		.WORD	0
0000	00AB2		.WORD	0
00000000	00AB4		.LONG	0
00000000	00AB8		.LONG	0
00000000	00ABC		.LONG	0
00000000	00AC0		.LONG	0
00	00AC4		.BYTE	0
00	00AC5		.BYTE	0
00	00AC6		.BYTE	0
00	00AC7		.BYTE	0
00000000	00AC8		.LONG	0
00000000	00ACC		.ADDRESS	INFAB
00000000	00AD0		.LONG	0
03	00AD4	OUTFAB:	.BYTE	3
50	00AD5		.BYTE	80
0000	00AD6		.WORD	0
00000000	00AD8		.LONG	0
00000000	00ADC		.LONG	0
00000000	00AE0		.LONG	0
00000000	00AE4		.LONG	0
0000	00AE8		.WORD	0
01	00AEA		.BYTE	1
00	00AEB		.BYTE	0
00000000	00AEC		.LONG	0
00	00AF0		.BYTE	0
00	00AF1		.BYTE	0
02	00AF2		.BYTE	2
02	00AF3		.BYTE	2
00000000	00AF4		.LONG	0
00000000	00AF8		.LONG	0
00000000	00AFC		.LONG	0
00000000	00B00		.ADDRESS	P.ABZ
00000000	00B04		.LONG	0
0A	00B08		.BYTE	10
00	00B09		.BYTE	0
0000	00B0A		.WORD	0
00000000	00B0C		.LONG	0
0000	00B10		.WORD	0
00	00B12		.BYTE	0
00	00B13		.BYTE	0
00000000	00B14		.LONG	0
00000000	00B18		.LONG	0
0000	00B1C		.WORD	0
00	00B1E		.BYTE	0
00	00B1F		.BYTE	0
00000000	00B20		.LONG	0
02	00B24	LSTNAM:	.BYTE	2

60	00B25	.BYTE	96
00	00B26	.BYTE	0
00	00B27	.BYTE	0
00000000	00B28	.LONG	0
00	00B2C	.BYTE	0
00	00B2D	.BYTE	0
00	00B2E	.BYTE	0
00	00B2F	.BYTE	0
00000000	00B30	.LONG	0
00000000	00B34	.LONG	0
0000#	00B38	.WORD	0[8]
0000#	00B48	.WORD	0[3]
0000#	00B4E	.WORD	0[3]
00000000	00B54	.LONG	0
00000000	00B58	.LONG	0
00	00B5C	.BYTE	0
00	00B5D	.BYTE	0
00	00B5E	.BYTE	0
00	00B5F	.BYTE	0
00	00B60	.BYTE	0
00	00B61	.BYTE	0
00#	00B62	.BYTE	0[2]
00000000	00B64	.LONG	0
00000000	00B68	.LONG	0
00000000	00B6C	.LONG	0
00000000	00B70	.LONG	0
00000000	00B74	.LONG	0
00000000	00B78	.LONG	0
00000000#	00B7C	.LONG	0[2]
13	00B84	.BYTE	19
58	00B85	.BYTE	88
0000	00B86	.WORD	0
00000000	00B88	.LONG	0
FC44	00B8C	.WORD	-956
00	00B8E	.BYTE	0
00	00B8F	.BYTE	0
0000 0000	00B90	.WORD	0, 0
00	00B94	.BYTE	0
00	00B95	.BYTE	0
0000	00B96	.WORD	0
00000000	00B98	.LONG	0
00000000	00B9C	.LONG	0
0000	00BA0	.WORD	0
0000	00BA2	.WORD	0
00000000	00BA4	.LONG	0
00000000	00BA8	.LONG	0
	00BAC	.BLKB	48
03	00BDC	.BYTE	3
50	00BDD	.BYTE	80
0000	00BDE	.WORD	0
00000000	00BE0	.LONG	0
00000000	00BE4	.LONG	0
00000000	00BE8	.LONG	0
00000000	00BEC	.LONG	0
0000	00BF0	.WORD	0
01	00BF2	.BYTE	1
20	00BF3	.BYTE	32

LSTPRO:

LSTFAB:

.....



```

00000000 00BF4 .LONG 0
00000000 00BF8 .BYTE 0
00000000 00BF9 .BYTE 0
00000000 00BFA .BYTE 2
00000000 00BFB .BYTE 2
00000000 00BFC .LONG 0
00000000 00C00 .ADDRESS LSTPRO
00000000 00C04 .ADDRESS LSTNAM
00000000 00C08 .ADDRESS P.ACA
00000000 00C0C .LONG 0
00000000 00C10 .BYTE 10
00000000 00C11 .BYTE 0
00000000 00C12 .WORD 132
00000000 00C14 .LONG 0
00000000 00C18 .WORD 0
00000000 00C1A .BYTE 0
00000000 00C1B .BYTE 0
00000000 00C1C .LONG 0
00000000 00C20 .LONG 0
00000000 00C24 .WORD 0
00000000 00C26 .BYTE 0
00000000 00C27 .BYTE 0
00000000 00C28 .LONG 0
00000000 00C2C LSTRAB: .BYTE 1
00000000 00C2D .BYTE 68
00000000 00C2E .WORD 0
00000000 00C30 .LONG 0
00000000 00C34 .LONG 0
00000000 00C38 .LONG 0
00000000 00C3C .WORD 0[3]
00000000 00C42 .WORD 0
00000000 00C44 .LONG 0
00000000 00C48 .WORD 0
00000000 00C4A .BYTE 0
00000000 00C4B .BYTE 0
00000000 00C4C .WORD 0
00000000 00C4E .WORD 0
00000000 00C50 .LONG 0
00000000 00C54 .ADDRESS DISBUF
00000000 00C58 .LONG 0
00000000 00C5C .LONG 0
00000000 00C60 .BYTE 0
00000000 00C61 .BYTE 0
00000000 00C62 .BYTE 0
00000000 00C63 .BYTE 0
00000000 00C64 .LONG 0
00000000 00C68 .ADDRESS LSTFAB
00000000 00C6C .LONG 0
00000000 00C70 NLSTNAM: .BYTE 2
00000000 00C71 .BYTE 96
00000000 00C72 .BYTE 0
00000000 00C73 .BYTE 0
00000000 00C74 .LONG 0
00000000 00C78 .BYTE 0
00000000 00C79 .BYTE 0
00000000 00C7A .BYTE 0
00000000 00C7B .BYTE 0

```

00000000	00C7C	.LONG	0
00000000	00C80	.LONG	0
0000#	00C84	.WORD	0[8]
0000#	00C94	.WORD	0[3]
0000#	00C9A	.WORD	0[3]
00000000	00CA0	.LONG	0
00000000	00CA4	.LONG	0
00	00CA8	.BYTE	0
00	00CA9	.BYTE	0
00	00CAA	.BYTE	0
00	00CAB	.BYTE	0
00	00CAC	.BYTE	0
00	00CAD	.BYTE	0
00#	00CAE	.BYTE	0[2]
00000000	00CB0	.LONG	0
00000000	00CB4	.LONG	0
00000000	00CB8	.LONG	0
00000000	00CBC	.LONG	0
00000000	00CC0	.LONG	0
00000000	00CC4	.LONG	0
00000000#	00CC8	.LONG	0[2]
13	00CD0	NLSTPRO: .BYTE	19
58	00CD1	.BYTE	88
0000	00CD2	.WORD	0
00000000	00CD4	.LONG	0
FC44	00CD8	.WORD	-956
00	00CDA	.BYTE	0
00	00CDB	.BYTE	0
0000 0000	00CDC	.WORD	0, 0
00	00CE0	.BYTE	0
00	00CE1	.BYTE	0
0000	00CE2	.WORD	0
00000000	00CE4	.LONG	0
00000000	00CE8	.LONG	0
0000	00CEC	.WORD	0
0000	00CEE	.WORD	0
00000000	00CF0	.LONG	0
00000000	00CF4	.LONG	0
	00CF8	.BLKB	48
03	00D28	NLSTFAB: .BYTE	3
50	00D29	.BYTE	80
0000	00D2A	.WORD	0
00000000	00D2C	.LONG	0
00000000	00D30	.LONG	0
00000000	00D34	.LONG	0
00000000	00D38	.LONG	0
0000	00D3C	.WORD	0
01	00D3E	.BYTE	1
20	00D3F	.BYTE	32
00000000	00D40	.LONG	0
00	00D44	.BYTE	0
00	00D45	.BYTE	0
02	00D46	.BYTE	2
02	00D47	.BYTE	2
00000000	00D48	.LONG	0
00C700000	00D4C	.ADDRESS	NLSTPRO
00000000	00D50	.ADDRESS	NLSTNAM

00000000	00D54	.ADDRESS P.ACB	
00000000	00D58	.LONG	0
0A	00D5C	.BYTE	10
00	00D5D	.BYTE	0
0084	00D5E	.WORD	132
00000000	00D60	.LONG	0
0000	00D64	.WORD	0
00	00D66	.BYTE	0
00	00D67	.BYTE	0
00000000	00D68	.LONG	0
00000000	00D6C	.LONG	0
0000	00D70	.WORD	0
00	00D72	.BYTE	0
00	00D73	.BYTE	0
00000000	00D74	.LONG	0
01	00D78	NLSTRAB: .BYTE	1
44	00D79	.BYTE	68
0000	00D7A	.WORD	0
00000000	00D7C	.LONG	0
00000000	00D80	.LONG	0
00000000	00D84	.LONG	0
0000	00D88	.WORD	0[3]
0000	00D8E	.WORD	0
00000000	00D90	.LONG	0
0000	00D94	.WORD	0
00	00D96	.BYTE	0
00	00D97	.BYTE	0
0000	00D98	.WORD	0
0000	00D9A	.WORD	0
00000000	00D9C	.LONG	0
00000000	00DA0	.ADDRESS DISBUF	
00000000	00DA4	.LONG	0
00000000	00DA8	.LONG	0
00	00DAC	.BYTE	0
00	00DAD	.BYTE	0
00	00DAE	.BYTE	0
00	00DAF	.BYTE	0
00000000	00DB0	.LONG	0
00000000	00DB4	.ADDRESS NLSTFAB	
00000000	00DB8	.LONG	0
15	00DBC	UAFKEY2: .BYTE	21
4C	00DBD	.BYTE	76
0000	00DBE	.WORD	0
00000000	00DC0	.LONG	0
00	00DC4	.BYTE	0
00	00DC5	.BYTE	0
00	00DC6	.BYTE	0
00	00DC7	.BYTE	0
00	00DC8	.BYTE	0
00	00DC9	.BYTE	0
00000000	00DCA	.LONG	0
03	00DCE	.BYTE	3
02	00DCF	.BYTE	2
00	00DD0	.BYTE	0
00	00DD1	.BYTE	0
00	00DD2	.BYTE	0
02	00DD3	.BYTE	2



0000	00DD4	.WORD	0
0000	00DD6	.WORD	0
0000	00DD8	.WORD	0
0024	00DDA	.WORD	36
0000	00DDC	.WORD	0
0000	00DDE	.WORD	0
0000	00DE0	.WORD	0
0000	00DE2	.WORD	0
0000	00DE4	.WORD	0
0000	00DE6	.WORD	0
0000	00DE8	.WORD	0
02	00DEA	.BYTE	20
00	00DEB	.BYTE	0
00	00DEC	.BYTE	0
00	00DED	.BYTE	0
00	00DEE	.BYTE	0
00	00DEF	.BYTE	0
00	00DF0	.BYTE	0
00	00DF1	.BYTE	0
0000	00DF2	.WORD	0
00000000	00DF4	.LONG	0
00000000	00DF8	.LONG	0
00	00DFC	.BYTE	0
00	00DFD	.BYTE	0
00	00DFE	.BYTE	0
00	00DFF	.BYTE	0
00	00E00	.BYTE	0
00	00E01	.BYTE	0
00	00E02	.BYTE	0
00	00E03	.BYTE	0
00	00E04	.BYTE	0
00	00E05	.BYTE	0
15	00E06	.BLKB	21
4C	00E08	.BYTE	76
0000	00E09	.BYTE	0
00000000	00E0A	.WORD	0
00000000	00E0C	.ADDRESS	UAFKEY2
00	00E10	.BYTE	0
00	00E11	.BYTE	0
00	00E12	.BYTE	0
00	00E13	.BYTE	0
00	00E14	.BYTE	0
00	00E15	.BYTE	0
00000000	00E16	.LONG	0
03	00E1A	.BYTE	3
04	00E1B	.BYTE	4
00	00E1C	.BYTE	0
00	00E1D	.BYTE	0
00	00E1E	.BYTE	0
01	00E1F	.BYTE	1
0000	00E20	.WORD	0
0000	00E22	.WORD	0
0000	00E24	.WORD	0
0024	00E26	.WORD	36
0000	00E28	.WORD	0
0000	00E2A	.WORD	0
0000	00E2C	.WORD	0

UAFKEY1:

UAFKEY2

0000	00E2E	.WORD	0
0000	00E30	.WORD	00
0000	00E32	.WORD	00
0000	00E34	.WORD	00
00	00E36	.BYTE	00
00	00E37	.BYTE	00
00	00E38	.BYTE	00
00	00E39	.BYTE	00
00	00E3A	.BYTE	00
00	00E3B	.BYTE	00
00	00E3C	.BYTE	00
00	00E3D	.BYTE	00
0000	00E3E	.WORD	00
00000000	00E40	.LONG	00
00000000	00E44	.LONG	00
00	00E48	.BYTE	00
00	00E49	.BYTE	00
00	00E4A	.BYTE	00
00	00E4B	.BYTE	00
00	00E4C	.BYTE	00
00	00E4D	.BYTE	00
00	00E4E	.BYTE	00
00	00E4F	.BYTE	00
00	00E50	.BYTE	00
00	00E51	.BYTE	00
00	00E52	.BLKB	21
15	00E54	.BYTE	76
4C	00E55	.BYTE	0
0000	00E56	.WORD	0
00000000	00E58	.ADDRESS	UAFKEY1
00	00E5C	.BYTE	0
00	00E5D	.BYTE	00
00	00E5E	.BYTE	00
00	00E5F	.BYTE	00
00	00E60	.BYTE	00
00	00E61	.BYTE	00
00000000	00E62	.LONG	00
00	00E66	.BYTE	00
00	00E67	.BYTE	00
00	00E68	.BYTE	00
00	00E69	.BYTE	00
00	00E6A	.BYTE	00
00	00E6B	.BYTE	00
0000	00E6C	.WORD	00
0000	00E6E	.WORD	00
0000	00E70	.WORD	00
0004	00E72	.WORD	00
0000	00E74	.WORD	00
0000	00E76	.WORD	00
0000	00E78	.WORD	00
0000	00E7A	.WORD	00
0000	00E7C	.WORD	00
0000	00E7E	.WORD	00
0000	00E80	.WORD	00
20	00E82	.BYTE	00
00	00E83	.BYTE	00
00	00E84	.BYTE	00

00	00E85	.BYTE	0
00	00E86	.BYTE	0
00	00E87	.BYTE	0
00	00E88	.BYTE	0
00	00E89	.BYTE	0
0000	00E8A	.WORD	0
00000000	00E8C	.LONG	0
00000000	00E90	.LONG	0
00	00E94	.BYTE	0
00	00E95	.BYTE	0
00	00E96	.BYTE	0
00	00E97	.BYTE	0
00	00E98	.BYTE	0
00	00E99	.BYTE	0
00	00E9A	.BYTE	0
00	00E9B	.BYTE	0
00	00E9C	.BYTE	0
00	00E9D	.BYTE	0
	00E9E	.BLKB	2
13	00EA0	UAFPRO: .BYTE	19
58	00EA1	.BYTE	88
0000	00EA2	.WORD	0
00000000	00EA4	.ADDRESS	UAFKEY0
FF00	00EA8	.WORD	-256
00	00EAA	.BYTE	0
00	00EAB	.BYTE	0
0000 0000	00EAC	.WORD	0, 0
00	00EB0	.BYTE	0
00	00EB1	.BYTE	0
0000	00EB2	.WORD	0
00000000	00EB4	.LONG	0
00000000	00EB8	.LONG	0
0000	00EBC	.WORD	0
0000	00EBE	.WORD	0
00000000	00EC0	.LONG	0
00000000	00EC4	.LONG	0
	00EC8	.BLKB	48
03	00EF8	UAFFAB: .BYTE	3
50	00EF9	.BYTE	80
0000	00EFA	.WORD	0
02000000	00EFC	.LONG	33554432
00000000	00F00	.LONG	0
00000000	00F04	.LONG	0
0000000A	00F08	.LONG	10
000A	00F0C	.WORD	10
0F	00F0E	.BYTE	15
0F	00F0F	.BYTE	15
00000000	00F10	.LONG	0
00	00F14	.BYTE	0
20	00F15	.BYTE	32
00	00F16	.BYTE	0
02	00F17	.BYTE	2
00000000	00F18	.LONG	0
00000000	00F1C	.ADDRESS	UAFPRO
00000000	00F20	.LONG	0
00000000	00F24	.ADDRESS	P.ACC
00000000	00F28	.ADDRESS	P.ACD



06	00F2C	.BYTE	6
04	00F2D	.BYTE	4
0584	00F2E	.WORD	1412
00000000	00F30	.LONG	00
0000	00F34	.WORD	00
00	00F36	.BYTE	00
00	00F37	.BYTE	00
00000000	00F38	.LONG	00
00000000	00F3C	.LONG	00
0000	00F40	.WORD	00
00	00F42	.BYTE	00
00	00F43	.BYTE	00
00000000	00F44	.LONG	00
15	00F48	.BYTE	21
4C	00F49	.BYTE	76
0000	00F4A	.WORD	00
00000000	00F4C	.LONG	00
00	00F50	.BYTE	00
00	00F51	.BYTE	00
00	00F52	.BYTE	00
00	00F53	.BYTE	00
00	00F54	.BYTE	00
00	00F55	.BYTE	00
00000000	00F56	.LONG	00
03	00F5A	.BYTE	33
00	00F5B	.BYTE	00
00	00F5C	.BYTE	00
00	00F5D	.BYTE	00
00	00F5E	.BYTE	00
01	00F5F	.BYTE	10
0000	00F60	.WORD	00
0000	00F62	.WORD	00
0000	00F64	.WORD	00
0040	00F66	.WORD	64
0000	00F68	.WORD	00
0000	00F6A	.WORD	00
0000	00F6C	.WORD	00
0000	00F6E	.WORD	00
0000	00F70	.WORD	00
0000	00F72	.WORD	00
0000	00F74	.WORD	00
20	00F76	.BYTE	32
00	00F77	.BYTE	00
00	00F78	.BYTE	00
00	00F79	.BYTE	00
00	00F7A	.BYTE	00
00	00F7B	.BYTE	00
00	00F7C	.BYTE	00
00	00F7D	.BYTE	00
0000	00F7E	.WORD	00
00000000	00F80	.LONG	00
00000000	00F84	.LONG	00
00	00F88	.BYTE	00
00	00F89	.BYTE	00
00	00F8A	.BYTE	00
00	00F8B	.BYTE	00
00	00F8C	.BYTE	00

NAFKEY1:

00	00F8D	.BYTE	0
00	00F8E	.BYTE	0
00	00F8F	.BYTE	0
00	00F90	.BYTE	0
00	00F91	.BYTE	0
00	00F92	.BLKB	2
15	00F94	NAFKEY0: .BYTE	21
4C	00F95	.BYTE	76
0000	00F96	.WORD	0
00000000	00F98	.ADDRESS NAFKEY1	
00	00F9C	.BYTE	0
00	00F9D	.BYTE	0
00	00F9E	.BYTE	0
00	00F9F	.BYTE	0
00	00FA0	.BYTE	0
00	00FA1	.BYTE	0
00000000	00FA2	.LONG	0
00	00FA6	.BYTE	0
00	00FA7	.BYTE	0
00	00FA8	.BYTE	0
00	00FA9	.BYTE	0
00	00FAA	.BYTE	0
00	00FAB	.BYTE	0
0000	00FAC	.WORD	0
0000	00FAE	.WORD	0
0000	00FB0	.WORD	0
0000	00FB2	.WORD	0
0000	00FB4	.WORD	0
0000	00FB6	.WORD	0
0000	00FB8	.WORD	0
0000	00FBA	.WORD	0
0000	00FBC	.WORD	0
0000	00FBE	.WORD	0
0000	00FC0	.WORD	0
40	00FC2	.BYTE	64
00	00FC3	.BYTE	0
00	00FC4	.BYTE	0
00	00FC5	.BYTE	0
00	00FC6	.BYTE	0
00	00FC7	.BYTE	0
00	00FC8	.BYTE	0
00	00FC9	.BYTE	0
0000	00FCA	.WORD	0
00000000	00FCC	.LONG	0
00000000	00FD0	.LONG	0
00	00FD4	.BYTE	0
00	00FD5	.BYTE	0
00	00FD6	.BYTE	0
00	00FD7	.BYTE	0
00	00FD8	.BYTE	0
00	00FD9	.BYTE	0
00	00FDA	.BYTE	0
00	00FDB	.BYTE	0
00	00FDC	.BYTE	0
00	00FDD	.BYTE	0
00	00FDE	.BLKB	2
13	00FE0	NAFPRO: .BYTE	19

```

      58 00FE1 .BYTE 88
      0000 00FE2 .WORD 0
      00000000 00FE4 .ADDRESS NAFKEY0
      FF00 00FE8 .WORD -256
      00 00FEA .BYTE 0
      00 00FEB .BYTE 0
      0000 0000 00FEC .WORD 0, 0
      00 00FF0 .BYTE 0
      00 00FF1 .BYTE 0
      0000 00FF2 .WORD 0
      00000000 00FF4 .LONG 0
      00000000 00FF8 .LONG 0
      0000 00FFC .WORD 0
      0000 00FFE .WORD 0
      00000000 01000 .LONG 0
      00000000 01004 .LONG 0
      01008 .BLKB 48
      03 01038 NAFFAB: .BYTE 3
      50 01039 .BYTE 80
      0000 0103A .WORD 0
      02000000 0103C .LONG 33554432
      00000000 01040 .LONG 0
      00000000 01044 .LONG 0
      0000000A 01048 .LONG 10
      000A 0104C .WORD 10
      0F 0104E .BYTE 15
      0F 0104F .BYTE 15
      00000000 01050 .LONG 0
      00 01054 .BYTE 0
      20 01055 .BYTE 32
      00 01056 .BYTE 0
      01 01057 .BYTE 1
      00000000 01058 .LONG 0
      00000000 0105C .ADDRESS NAFPRO
      00000000 01060 .LONG 0
      00000000 01064 .ADDRESS P.ACE
      00000000 01068 .ADDRESS P.ACF
      06 0106C .BYTE 6
      04 0106D .BYTE 4
      0064 0106E .WORD 100
      00000000 01070 .LONG 0
      0000 01074 .WORD 0
      00 01076 .BYTE 0
      00 01077 .BYTE 0
      00000000 01078 .LONG 0
      00000000 0107C .LONG 0
      0000 01080 .WORD 0
      00 01082 .BYTE 0
      00 01083 .BYTE 0
      00000000 01084 .LONG 0
      01088 STATUS: .BLKB 4
      .PSECT $GLOBAL$,NOEXE,2
      00000 00000 DISBUF:: .BLKB 132
      00000084 00084 DISDSC:: .LONG 132
      00000000 00088 .ADDRESS DISBUF

```

```

0008C FAODSC::.BLKB 8
00094 RABPTR::.BLKB 4
00098 UAF$GQ_SYSAFF::.BLKB 8
000A0 UAF$GL_CTLMSK::.BLKB 8
00000000 000A8 BY_ACCOUNT::.LONG 0
000AC MATCH_TOKEN::.BLKB 66
000EE .BLKB 2
000F0 MATCH_TOKENLEN::.BLKB 4
000F4 WILD_NETUSER::.BLKB 4
00000000 000F8 CALL_COUNT::.LONG 0
00# 000FC TOKENDSC::.BYTE 0[3]
02 000FF .BYTE 2
00100 .BLKB 4
00104 CMDLINDSC::.BLKB 8
0010C NETUAF_EXISTS::.BLKB 4
00110 RDB_EXISTS::.BLKB 4
00114 RMSERR::.BLKB 4
00118 RIGHTSLIST_MODIFIED::.BLKB 1
00119 .BLKB 3
0011C RECBUF::.BLKB 1412
006A0 NETBUF::.BLKB 100
01 00704 UAFRAB::.BYTE 1
44 00705 .BYTE 68
0000 00706 .WORD 0
00000000 00708 .LONG 0
00000000 0070C .LONG 0
00000000 00710 .LONG 0
0000# 00714 .WORD 0[3]
0000 0071A .WORD 0
00000000 0071C .LONG 0
0000 00720 .WORD 0
00 00722 .BYTE 0
00 00723 .BYTE 0
0584 00724 .WORD 1412
0000 00726 .WORD 0
00000000 00728 .LONG 0
00000000 0072C .LONG 0
00000000 00730 .LONG 0
00000000 00734 .ADDRESS RECBUF+4
20 00738 .BYTE 32
00 00739 .BYTE 0
00 0073A .BYTE 0
00 0073B .BYTE 0
00000000 0073C .LONG 0
00000000 00740 .ADDRESS UAFFAB

```



```
00000000 00744 .LONG 0
      01 00748 NAFRAB: .BYTE 1
      44 00749 .BYTE 68
      0000 0074A .WORD 0
00000000 0074C .LONG 0
00000000 00750 .LONG 0
00000000 00754 .LONG 0
      0000# 00758 .WORD 0[3]
      0000 0075E .WORD 0
00000000 00760 .LONG 0
      0000 00764 .WORD 0
      01 00766 .BYTE 1
      00 00767 .BYTE 0
      0064 00768 .WORD 100
      0064 0076A .WORD 100
00000000 0076C .LONG 0
00000000 00770 .LONG 0
00000000 00774 .LONG 0
00000000 00778 .ADDRESS NETBUF
      40 0077C .BYTE 64
      00 0077D .BYTE 0
      00 0077E .BYTE 0
      00 0077F .BYTE 0
00000000 00780 .LONG 0
00000000 00784 .ADDRESS NAFFAB
00000000 00788 .LONG 0
      0078C TIME_BUF: .BLKB 8
      00794 PWD_FLAG: .BLKB 4
      01 00798 OUTRAB: .BYTE 1
      44 00799 .BYTE 68
      0000 0079A .WORD 0
00000000 0079C .LONG 0
00000000 007A0 .LONG 0
00000000 007A4 .LONG 0
      0000# 007A8 .WORD 0[3]
      0000 007AE .WORD 0
00000000 007B0 .LONG 0
      0000 007B4 .WORD 0
      00 007B6 .BYTE 0
      00 007B7 .BYTE 0
      0000 007B8 .WORD 0
      0000 007BA .WORD 0
00000000 007BC .LONG 0
00000000 007C0 .ADDRESS DISBUF
00000000 007C4 .LONG 0
00000000 007C8 .LONG 0
      00 007CC .BYTE 0
      00 007CD .BYTE 0
      00 007CE .BYTE 0
      00 007CF .BYTE 0
00000000 007D0 .LONG 0
00000000 007D4 .ADDRESS OUTFAB
00000000 007D8 .LONG 0
      007DC FOUND_MATCH: .BLKB 4
```

007E0 UIC\_FLAG::  
                  .BLKB 4  
007E4 GRP\_WILD::  
                  .BLKB 4  
007E8 MEM\_WILD::  
                  .BLKB 4  
007EC STR\_WILD::  
                  .BLKB 4

SD\_TOKEN1= P.AAA  
SD\_TOKEN2= P.AAC  
SD\_BRIEF= P.AAE  
SD\_FULL= P.AAG  
SD\_ADD\_IDENTIFIER= P.AAI  
SD\_REMOVE\_IDENTIFIER= P.AAK  
SD\_MODIFY\_IDENTIFIER= P.AAM  
ENCRYPT== 2  
DISBUFLN== 132  
SYSUAF\_STRING= P.AAP  
NETUAF\_STRING= P.AAQ  
MOD\_ACT\_DSC= P.AAR  
ADD\_ACT\_DSC= P.AAT  
REM\_ACT\_DSC= P.AAV  
FAO\_LIN\_DSC= P.AAX  
DBL\_COLON= P.AAZ  
WAKEDELTA= P.ABB  
DEFUSER= P.ABC  
DEFPASS= P.ABD  
DEFCLITABL= P.ABE  
DEFACT= P.ABF  
DEFCLI= P.ABG  
DEFOWNER= P.ABH  
DEFLGICMD= P.ABI  
DEFGRP= 128  
DEFMEM= 128  
DEFDIR= P.ABJ  
DEFDEV= P.ABK  
DEFBIOLM= 6  
DEFBYTLM= 4096  
DEFDIOLM= 6  
DEFFILLM= 20  
DEFFLAGS= 0  
DEFTQCNT= 10  
DEFPRCNT= 2  
DEFPRI= 4  
DEFQUEPRI= 4  
DEFWSQUOTA= 200  
DEFDFWSCNT= 150  
DEFWSEXTENT= 500  
DEFPCPUTIM= 0  
DEFASTLM= 10  
DEFPGFLQUOTA= 10000  
DEFENQLM= 10  
DEFPBYTLM= 0  
DEFSHRFILLM= 0

```

DEFMAXJOBS= 0
DEFMAXACCTJOBS= 0
DEFMAXDETACH= 0
DEFJTQUOTA= 1024
DEFPRIMEDAYS= 96
DEFHOURS= 0
DEFPRIV= P.ABL
DEFPWDLENGTH= 6
DEFPWDLIFE= P.ABM
SYSUSER= P.ABN
SYSPASS= P.ABO
SYSCLITABL= P.ABP
SYSACT= P.ABQ
SYSCLI= P.ABR
SYSOWNER= P.ABS
SYSLGICMD= P.ABT
SYSGRP= 1
SYSTEM= 4
SYSDIR= P.ABU
SYSDEV= P.ABV
SYSBIOLM= 12
SYSBYTLM= 20480
SYSDIOLM= 12
SYSFILLM= 20
SYSFLAGS= 0
SYSTQCNT= 20
SYSPRCNT= 10
SYSPRI= 4
SYSQUEPRI= 4
SYSWSQUOTA= 350
SYSWSEXTENT= 1024
SYSDFWSCNT= 150
SYSCPUTIM= 0
SYSASTLM= 20
SYSPGFLQUOTA= 10000
SYSENQLM= 20
SYSPBYTLM= 0
SYSSHRFILLM= 0
SYSMAXJOBS= 0
SYSMAXDETACH= 0
SYSJTQUOTA= 1024
SYSMAXACCTJOBS= 0
SYSPRIMEDAYS= 96
SYSHOURS= 0
SYSPRIV= P.ABW
SYSPWDLENGTH= 8
SYSPWDLIFE= P.ABX
TOKENLEN== TOKENDSC
TOKENPTR== TOKENDSC+4
REC_USER_DSC== P.ACG
REC_ENCRYPT_DSC== P.ACH
SYMBOL_STR== P.ACI
ACCPMPT= P.ACJ
ACCPMPT2= P.ACK
NEUMSG20= P.ACL
FOREIGN_CMDLINDSC= P.ACM
      .EXTRN CLIS_BUFOVF, CLIS_NOCLINT

```

```
.EXTRN LBR$OUTPUT_HELP
.EXTRN LIB$GET_FOREIGN
.EXTRN LIB$GET_INPUT, LIB$PUT_OUTPUT
.EXTRN FMG$MATCH_NAME, CLISDCC_PARSE
.EXTRN CLISDISPATCH, CLISPRESNT
.EXTRN CLISGET_VALUE, UPDATE_RECORD
.EXTRN PARSE_WILD, LGISHPWD
.EXTRN UAF$ADD_IDENT_RECBUF
.EXTRN UAF$BUILT_HOLDER
.EXTRN UAF$FIND_OIC, UAF$REMOVE_IDENT_RECBUF
.EXTRN UAF$WRITE_RIGHTS
.EXTRN EXESGL_SYSUIC, RDB_HEADER_FLAG
.EXTRN RDB_LIST_FLAG, ATTRIBUTES
.EXTRN HOLDER, IDENT, AUTHORIZE_COMMANDS
.EXTRN PRVSAB_NAMES, LIB$SIGNAL
.EXTRN UAF$_ADDERR, UAF$_ADDMSG
.EXTRN UAF$_BADNODFORM
.EXTRN UAF$_BADSPC, UAF$_BADUSR
.EXTRN UAF$_CLIWARNMSG
.EXTRN UAF$_CMDTOOLONG
.EXTRN UAF$_CONERR, UAF$_COPMSG
.EXTRN UAF$_CREERR, UAF$_DEFERR
.EXTRN UAF$_DEFPWD, UAF$_DONEMSG
.EXTRN UAF$_GETERR, UAF$_HELPERR
.EXTRN UAF$_INVCMD, UAF$_INVRSP
.EXTRN UAF$_INVUSERNAME
.EXTRN UAF$_KEYNOTFND, UAF$_KEYNOTUNG
.EXTRN UAF$_LSTERR, UAF$_LSTMSG1
.EXTRN UAF$_LSTMSG2, UAF$_MDFYERR
.EXTRN UAF$_MDFYMSG, UAF$_NAFADDERR
.EXTRN UAF$_NAFADDMSG, UAF$_NAFAEX
.EXTRN UAF$_NAFCONERR, UAF$_NAFCREERR
.EXTRN UAF$_NAFDNE, UAF$_NAFDONEMSG
.EXTRN UAF$_NAFNOMODS, UAF$_NAFUAEERR
.EXTRN UAF$_NAMETOOBIG
.EXTRN UAF$_NETLSTMSG, UAF$_NAOFIL
.EXTRN UAF$_NAONAF, UAF$_NOARG
.EXTRN UAF$_NODEFPWD, UAF$_NODTOOBIG
.EXTRN UAF$_NOMODS, UAF$_NOTUNG
.EXTRN UAF$_NOUSERNAME
.EXTRN UAF$_PREMMSG, UAF$_PUTERR
.EXTRN UAF$_RDBDONEMSG
.EXTRN UAF$_RDBMDFYERR
.EXTRN UAF$_RDBMDFYERRU
.EXTRN UAF$_RDBMDFYMSG
.EXTRN UAF$_RDBNOMODS, UAF$_REMDEF
.EXTRN UAF$_REMERR, UAF$_REMMSG
.EXTRN UAF$_REMSYS, UAF$_RENDEF
.EXTRN UAF$_RENMSG, UAF$_RENSYS
.EXTRN UAF$_RONLY, UAF$_SHOW_ERR
.EXTRN UAF$_SYSMSG1, UAF$_SYSMSG2
.EXTRN UAF$_UAEERR, UAF$_OICERR
.EXTRN UAF$_ZISQUAL, UAF$_ZZPRACREN
.EXTRN SYSS$OPEN, SYSS$CONNECT
.EXTRN SYSS$CREATE

.PSECT $CODE$,NOWRT,2
```



			OFFC	00000	START:	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11		0902
	5B	00000000G	00	9E	00002	MOVAB	AUTHORIZE COMMANDS, R11		
	5A	00000000G	00	9E	00009	MOVAB	SYSS\$CONNECT, R10		
	59	00000000G	8F	D0	00010	MOVL	#CLIS\$ NOCLINT, R9		
	58	00000000G	00	9E	00017	MOVAB	LIB\$STOP, R8		
	57	00000000G	00	9E	0001E	MOVAB	CMDLINDSC, R7		
	56	00000000G	00	9E	00025	MOVAB	STATUS, R6		
		14	A7	94	0002C	CLRB	RIGHTSLIST MODIFIED		0947
		EFC4	C6	7C	0002F	CLRQ	MODIFY_FLAG		0949
		F988	C6	9F	00033	PUSHAB	INFAB		0955
00000000G	00		01	FB	00037	CALLS	#1, SYSS\$OPEN		
	66		50	D0	0003E	MOVL	R0, STATUS		
10	A7		50	D0	00041	MOVL	R0, RMSERR		
	05		50	E8	00045	BLBS	R0, 1\$		
	68		66	DD	00048	PUSHL	STATUS		0957
		FA08	01	FB	0004A	CALLS	#1, LIB\$STOP		
	6A		C6	9F	0004D	PUSHAB	INRAB		0959
	66		01	FB	00051	CALLS	#1, SYSS\$CONNECT		
10	A7		50	D0	00054	MOVL	R0, STATUS		
	05		50	D0	00057	MOVL	R0, RMSERR		
			50	E8	0005B	BLBS	R0, 2\$		
	68		66	DD	0005E	PUSHL	STATUS		0961
		FA4C	01	FB	00060	CALLS	#1, LIB\$STOP		
00000000G	00		C6	9F	00063	PUSHAB	OUTFAB		0963
		0694	01	FB	00067	CALLS	#1, SYSS\$CREATE		
	6A		C7	9F	0006E	PUSHAB	OUTRAB		0964
00000000V	00		01	FB	00072	CALLS	#1, SYSS\$CONNECT		
			00	FB	00075	CALLS	#0, SETUP		0966
			57	DD	0007C	PUSHL	R7		0973
			7E	D4	0007E	CLRL	-(SP)		
00000000G	00	00000000G	00	9F	00080	PUSHAB	FOREIGN CMDLINDSC		
	30		03	FB	00086	CALLS	#3, LIB\$GET_FOREIGN		
			50	E9	0008D	BLBC	R0, 4\$		
			67	D5	00090	TSTL	CMDLINDSC		
			2C	13	00092	BEQL	4\$		
04	A7	F018	C6	9E	00094	MOVAB	CMDBUF, CMDLINDSC+4		0979
		0880	8F	BB	0009A	PUSHR	#^M<R7,R11>		0981
00000000G	00		02	FB	0009E	CALLS	#2, CLISDCL_PARSE		
	66		50	D0	000A5	MOVL	R0, STATUS		
	09		50	E9	000AB	BLBC	R0, 3\$		
00000000G	00		00	FB	000AB	CALLS	#0, CLISDISPATCH		0984
			43	11	000B2	BRB	7\$		0985
	59		66	D1	000B4	CMPL	STATUS, R9		0991
			37	13	000B7	BEQL	6\$		
00000000V	00		00	FB	000B9	CALLS	#0, ACC\$EXIT		0995
00000000V	00		00	FB	000C0	CALLS	#0, GET_CMD_LINE		1008
	F6		50	E9	000C7	BLBC	R0, 4\$		
		0880	8F	BB	000CA	PUSHR	#^M<R7,R11>		1010
00000000G	00		02	FB	000CE	CALLS	#2, CLISDCL_PARSE		
	66		50	D0	000D5	MOVL	R0, STATUS		
	10		50	E9	000D8	BLBC	R0, 5\$		
08	00		00	2C	000DB	MOVCS	#0, (SP), #0, #8, UAF\$GL_CTLMSK		1013
		9C	A7		000E0				
00000000G	00		00	FB	000E2	CALLS	#0, CLISDISPATCH		1014
			D5	11	000E9	BRB	4\$		1010
	59		66	D1	000EB	CMPL	STATUS, R9		1017

UAFMAIN  
V04-000

start - controlling code

G 13  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 39  
(3)

68

50

D0	12	000EE		BNEQ	4\$
59	DD	000F0	6\$:	PUSHL	R9
01	FB	000F2		CALLS	#1, LIB\$STOP
C9	11	000F5		BRB	4\$
01	D0	000F7	7\$:	MOVL	#1, R0
04	000FA			RET	

: 1019  
: 1010  
: 1022  
: 1024

: Routine Size: 251 bytes,      Routine Base: \$CODE\$ + 0000

UA  
VO

## setup - open initial files

```
933 1025 1 %sbttl 'setup - open initial files'
934 1026 1 routine setup : novalue =
935 1027 2 begin
936 1028 2
937 1029 2 ++
938 1030 2
939 1031 2 FUNCTIONAL DESCRIPTION:
940 1032 2
941 1033 2 This routine does all of the initial file manipulation for the
942 1034 2 program. It determines whether or not a previous SYSUAF.DAT exists.
943 1035 2 If not, it creates one (if the user wishes to proceed).
944 1036 2
945 1037 2 INPUTS:
946 1038 2
947 1039 2 none
948 1040 2
949 1041 2 IMPLICIT INPUTS:
950 1042 2
951 1043 2 none
952 1044 2
953 1045 2 OUTPUTS:
954 1046 2
955 1047 2 none
956 1048 2
957 1049 2 IMPLICIT OUTPUTS:
958 1050 2
959 1051 2 none
960 1052 2
961 1053 2 ROUTINE VALUE:
962 1054 2
963 1055 2 none
964 1056 2
965 1057 2 SIDE EFFECTS:
966 1058 2
967 1059 2 none
968 1060 2 --
969 1061 2
970 1062 2 local
971 1063 2 status : long,
972 1064 2 curpriv : vector [2],
973 1065 2 propriv : vector [2],
974 1066 2 item_list : block [64, byte],
975 1067 2 newFile; : indicates new file must
976 1068 2 : be created.
977 1069 2
978 1070 2 item_list [0.0.16.0] = 4;
979 1071 2 item_list [2.0.16.0] = jpi$_pid;
980 1072 2 item_list [4.0.32.0] = pid;
981 1073 2 item_list [8.0.32.0] = 0;
982 1074 2 item_list [12.0.16.0] = 12;
983 1075 2 item_list [14.0.16.0] = jpi$_username;
984 1076 2 item_list [16.0.32.0] = username_buf;
985 1077 2 item_list [20.0.32.0] = username_dsc[0];
986 1078 2 item_list [24.0.16.0] = 8;
987 1079 2 item_list [26.0.16.0] = jpi$_curpriv;
988 1080 2 item_list [28.0.32.0] = curpriv;
989 1081 2 item_list [32.0.32.0] = 0;
```

## setup - open initial files

```

990 1082 2 item_list [36.0,16.0] = 8;
991 1083 item_list [38.0,16.0] = jpi$_procpriv;
992 1084 item_list [40.0,32.0] = procpriv;
993 1085 item_list [44.0,32.0] = 0;
994 1086 item_list [48.0,16.0] = 4;
995 1087 item_list [50.0,16.0] = jpi$_sts;
996 1088 item_list [52.0,32.0] = pcb$_sts;
997 1089 item_list [56.0,32.0] = 0;
998 1090 item_list [60.0,32.0] = 0;
999 1091
1000 1092 $getjpi (itmlst = item_list);          ! Obtain pid, username and privs
1001 1093
1002 1094 username_dsc[0] = namelen (uaf$_username, username_buf);
1003 1095
1004 1096 *****
1005 1097
1006 1098                               Open SYSUAF.DAT
1007 1099
1008 1100 *****
1009 1101
1010 1102 newfile = false;                      ! note no new file yet
1011 1103
1012 1104 if rmsbad ($open (fab = uaffab))
1013 1105 then
1014 1106
1015 1107     sysuaf.dat doesn't exist
1016 1108
1017 1109     begin
1018 1110         LIB$SIGNAL(UAF$_NAOFIL, 0, .rmserr);
1019 1111         if .rmserr eql rms$_fnt
1020 1112         then
1021 1113             begin
1022 1114                 while true
1023 1115                 do
1024 1116
1025 1117                 Ask if a new one is desired
1026 1118
1027 1119                 begin
1028 1120                     ask (newmsg20, cmdbuf[0], cmdbuflen);
1029 1121                     if .cmdbuf [0] eql 'Y'
1030 1122                     then exitloop newfile = true;
1031 1123                     if .cmdbuf [0] eql 'N'
1032 1124                     then acc$exit ();
1033 1125                     LIB$SIGNAL(UAF$_INVRSP);
1034 1126                     end;
1035 1127                 end
1036 1128             else
1037 1129                 acc$exit ();
1038 1130             end;
1039 1131
1040 1132
1041 1133
1042 1134 The file will be created if it does not already exist.
1043 1135 In any case connect the RAB.
1044 1136
1045 1137
1046 1138 2 if .newfile
```



setup - open initial files

```
1047 1139 then
1048 1140
1049 1141 A new file is requested
1050 1142
1051 1143 if rmsbad ($create (fab = uaffab))
1052 1144 then
1053 1145
1054 1146 Quit regardless of error on a $CREATE: don't
1055 1147 want to give read-only user ability to create a file
1056 1148
1057 1149 LIB$SIGNAL(UAF$_CREERR, 0, .rmserr);
1058 1150
1059 1151 if rmsbad ($connect (rab = uafrab))
1060 1152 then LIB$SIGNAL(UAF$_CONERR, 0, .rmserr);
1061 1153 uafrab[rab$b_rac] = rab$c_key; ! normal access is by key
1062 1154
1063 1155
1064 1156 Check to see if there was no old file to use. If so write a default and
1065 1157 a system manager record.
1066 1158
1067 1159
1068 1160 if .newfile
1069 1161 then
1070 1162 begin
1071 1163 modify_flag = true; ! must rename when done
1072 1164 build_ini_recs (); ! build default and system manager records
1073 1165 uafrab[rab$w_rsz] = uaf$c_fixed;
1074 1166
1075 1167 default_size = uaf$c_fixed;
1076 1168 uafrab[rab$l_rbf] = default_record; ! insert default record address
1077 1169 if rmsbad ($put (rab = uafrab))
1078 1170 then LIB$SIGNAL(UAF$_PUTERR, 0, .rmserr); ! report error
1079 1171 uafrab[rab$l_rbf] = recbuf; ! establish proper address (and
1080 1172 ! address of system record)
1081 1173 if rmsbad ($put (rab = uafrab)) ! output system record
1082 1174 then LIB$SIGNAL(UAF$_PUTERR, 0, .rmserr);
1083 1175 end
1084 1176 else
1085 1177
1086 1178 Read in the default record.
1087 1179
1088 1180 begin
1089 1181 uafrab[rab$l_ubf] = default_record;
1090 1182 if not locate_user (.defuser<0,8>, defuser+1, false)
1091 1183 then LIB$SIGNAL(UAF$_DEFERR, 0, .rmserr);
1092 1184 default_size = .uafrab[rab$w_rsz];
1093 1185 end;
1094 1186
1095 1187 uafrab[rab$l_ubf] = recbuf; ! establish proper addresses
1096 1188 uafrab[rab$l_rbf] = recbuf;
1097 1189
1098 1190 *****
1099 1191
1100 1192 Open NETUAF.DAT
1101 1193
1102 1194 *****
1103 1195
```

setup - open initial files

```
1104 1196 2 netuaf_exists = true;           ! Assume NETUAF.DAT exists...
1105 1197
1106 1198
1107 1199  Try to open NETUAF.DAT and see what happens...
1108 1200
1109 1201  if rmsbad ($open (fab = naffab))
1110 1202  then
1111 1203
1112 1204  Couldn't open it
1113 1205
1114 1206  if .rmserr eql rms$_fnf
1115 1207  then
1116 1208  netuaf_exists = false           ! it doesn't exist
1117 1209  else
1118 1210  LIB$SIGNAL(UAF$_NAONAF, 0, .rmserr)    ! open error for some other reason
1119 1211
1120 1212  else
1121 1213  NETUAF.DAT opened without error
1122 1214
1123 1215
1124 1216  if rmsbad ($connect (rab = nafcab))
1125 1217  then LIB$SIGNAL(UAF$_NAFCONERR)      ! connect error
1126 1218
1127 1219  Everything opened and connected, establish proper NETUAF addresses
1128 1220
1129 1221  else
1130 1222  begin
1131 1223  nafcab [rab$_ubf] = netbuf;
1132 1224  nafcab [rab$_rbf] = netbuf;
1133 1225  end;
1134 1226
1135 1227
1136 1228  Check to see if the rights data base exists. Try to translate an ID
1137 1229  and if we get a file not found error then it doesn't exist.
1138 1230
1139 1231  ident[uic$_v_format] = uic$_k_uic_format ;
1140 1232  ident[uic$_v_group] = 1 ;
1141 1233  ident[uic$_v_member] = 4 ;
1142 1234  status = $idtoasc ( id = .ident ) ;
1143 1235  if .status eql rms$_fnf
1144 1236  then rdb_exists = false
1145 1237  else rdb_exists = true ;
1146 1238
1147 1239 1 end;
```

```
.EXTRN  SYSS$GETJPI, SYSS$PUT
.EXTRN  SYSS$IDTOASC
```

```
0FFC 00000 SETUP: .WORD  Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
5B 00000000G 00 9E 00002 MOVAB SYSS$PUT, R11
5A 00000000G 00 9E 00009 MOVAB SYSS$CONNECT, R10
59 00000000V 00 9E 00010 MOVAB ACC$EXIT, R6
58 00000000G 00 9E 00017 MOVAB SYSS$OPEN, R8
57 00000000G 00 9E 0001E MOVAB NEWMSG20, R7
56 00000000G 00 9E 00025 MOVAB IDENT, R6
```

: 1026

		55	00000000G	00	9E	0002C	MOVAB	LIB\$SIGNAL, R5		
		54	000000000	00	9E	00033	MOVAB	USERNAME_BUF, R4		
		53	000000000	00	9E	0003A	MOVAB	RMSERR, R3		
		5E	84	AE	9E	00041	MOVAB	-76(SP), SP		
			03190004	8F	DD	00045	PUSHL	#51970052		1070
	04	AE	24	A4	9E	0004B	MOVAB	PID, ITEM_LIST+4		1072
			08	AE	D4	00050	CLRL	ITEM_LIST+8		1073
	0C	AE	0202000C	8F	D0	00053	MOVL	#33685516, ITEM_LIST+12		1074
	10	AE		64	9E	0005B	MOVAB	USERNAME_BUF, ITEM_LIST+16		1076
	14	AE	28	A4	9E	0005F	MOVAB	USERNAME_DSC, ITEM_LIST+20		1077
	18	AE	04000008	8F	D0	00064	MOVL	#67108872, ITEM_LIST+24		1078
	1C	AE	48	AE	9E	0006C	MOVAB	CURPRIV, ITEM_LIST+28		1080
			20	AE	D4	00071	CLRL	ITEM_LIST+32		1081
	24	AE	02040008	8F	D0	00074	MOVL	#338T6584, ITEM_LIST+36		1082
	28	AE	40	AE	9E	0007C	MOVAB	PROCPRIV, ITEM_LIST+40		1084
			2C	AE	D4	00081	CLRL	ITEM_LIST+44		1085
	30	AE	03050004	8F	D0	00084	MOVL	#50659332, ITEM_LIST+48		1086
	34	AE	20	A4	9E	0008C	MOVAB	PCB_STS, ITEM_LIST+52		1088
			38	AE	7C	00091	CLRQ	ITEM_LIST+56		1089
				7E	7C	00094	CLRQ	-(SP)		1092
				7E	D4	00096	CLRL	-(SP)		
			0C	AE	9F	00098	PUSHAB	ITEM_LIST		
				7E	7C	0009B	CLRQ	-(SP)		
				7E	D4	0009D	CLRL	-(SP)		
		00000000G	00	07	FB	0009F	CALLS	#7, SYSS\$GETJPI		
	64		20	20	3A	000A6	LOCC	#32, #32, USERNAME_BUF		1094
28	A4		20	50	C3	000AA	SUBL3	R0, #32, USERNAME_DSC		
				52	D4	000AF	CLRL	NEWFILE		1102
				C4	9F	000B1	PUSHAB	UAF\$FAB		1104
			68	01	FB	000B5	CALLS	#1, SYSS\$OPEN		
			63	50	D0	000B8	MOVL	R0, RMSERR		
			4F	50	E8	000BB	BLBS	R0, 5\$		
				63	DD	000BE	PUSHL	RMSERR		1110
				7E	D4	000C0	CLRL	-(SP)		
		00000000G		8F	DD	000C2	PUSHL	#UAF\$ NAOFIL		
			65	03	FB	000C8	CALLS	#3, LIB\$SIGNAL		
	00018292		8F	63	D1	000CB	CMPL	RMSERR, #98962		1111
				36	12	000D2	BNEQ	4\$		
			7E	8F	3C	000D4	MOVZWL	#1024, -(SP)		1120
			0098	C4	9F	000D9	PUSHAB	CMDBUF		
				57	DD	000DD	PUSHL	R7		
	00000000V		00	03	FB	000DF	CALLS	#3, ASK		
			50	C4	9A	000E6	MOVZBL	CMDBUF, R0		1121
	59		8F	50	91	000EB	CMPL	R0, #89		
				05	12	000EF	BNEQ	2\$		
			52	01	D0	000F1	MOVL	#1, NEWFILE		1122
				17	11	000F4	BRB	5\$		
	4E		8F	50	91	000F6	CMPL	R0, #78		1123
				03	12	000FA	BNEQ	3\$		
			69	00	FB	000FC	CALLS	#0, ACC\$EXIT		1124
		00000000G		8F	DD	000FF	PUSHL	#UAF\$ INVRSP		1125
			65	01	FB	00105	CALLS	#1, LIB\$SIGNAL		
				CA	11	00108	BRB	1\$		1114
			69	00	FB	0010A	CALLS	#0, ACC\$EXIT		1129
			1E	52	E9	0010D	BLBC	NEWFILE, 6\$		1138
				C4	9F	00110	PUSHAB	UAF\$FAB		1143
	00000000G		00	01	FB	00114	CALLS	#1, SYSS\$CREATE		

63	50	D0	0011B	MOVL	R0, RMSERR	
0D	50	E8	0011E	BLBS	R0, 6\$	
	63	DD	00121	PUSHL	RMSERR	1149
	7E	D4	00123	CLRL	-(SP)	
00000000G	8F	DD	00125	PUSHL	#UAF\$ CREERR	
65	03	FB	0012B	CALLS	#3, LIB\$SIGNAL	
05F0	C3	9F	0012E	PUSHAB	UAFRAB	1151
6A	01	FB	00132	CALLS	#1, SYSS\$CONNECT	
63	50	D0	00135	MOVL	R0, RMSERR	
0D	50	E8	00138	BLBS	R0, 7\$	
	63	DD	0013B	PUSHL	RMSERR	1152
	7E	D4	0013D	CLRL	-(SP)	
00000000G	8F	DD	0013F	PUSHL	#UAF\$ CONERR	
65	03	FB	00145	CALLS	#3, LIB\$SIGNAL	
060E	01	90	00148	MOVB	#1, UAFRAB+30	1153
C3	52	E9	0014D	BLBC	NEWFILE, 9\$	1160
44	01	D0	00150	MOVL	#1, MODIFY FLAG	1163
00000000V	00	FB	00154	CALLS	#0, BUILD INI RECS	1164
0612	C3	8F	B0 0015B	MOVW	#644, UAFRAB+34	1165
0498	C4	8F	B0 00162	MOVW	#644, DEFAULT SIZE	1167
0618	C3	9E	00169	MOVAB	DEFAULT_RECORD, UAFRAB+40	1168
		9F	00170	PUSHAB	UAFRAB	1169
6B	01	FB	00174	CALLS	#1, SYSS\$PUT	
63	50	D0	00177	MOVL	R0, RMSERR	
0D	50	E8	0017A	BLBS	R0, 8\$	
	63	DD	0017D	PUSHL	RMSERR	1170
	7E	D4	0017F	CLRL	-(SP)	
00000000G	8F	DD	00181	PUSHL	#UAF\$ PUTERR	
65	03	FB	00187	CALLS	#3, LIB\$SIGNAL	
0618	A3	9E	0018A	MOVAB	RECBUF, UAFRAB+40	1171
	C3	9F	00190	PUSHAB	UAFRAB	1173
6B	01	FB	00194	CALLS	#1, SYSS\$PUT	
63	50	D0	00197	MOVL	R0, RMSERR	
3F	50	E8	0019A	BLBS	R0, 11\$	
	63	DD	0019D	PUSHL	RMSERR	1174
	7E	D4	0019F	CLRL	-(SP)	
00000000G	8F	DD	001A1	PUSHL	#UAF\$ PUTERR	
65	03	FB	001A7	CALLS	#3, LIB\$SIGNAL	
	30	11	001AA	BRB	11\$	1160
0614	C3	9E	001AC	MOVAB	DEFAULT_RECORD, UAFRAB+36	1181
	7E	D4	001B3	CLRL	-(SP)	1182
	C7	9F	001B5	PUSHAB	DEFUSER+1	
FEFB	C7	9A	001B9	MOVZBL	DEFUSER, -(SP)	
FEFA	03	FB	001BE	CALLS	#3, LOCATE_USER	
00000000V	50	E8	001C5	BLBS	R0, 10\$	
	63	DD	001C8	PUSHL	RMSERR	1183
	7E	D4	001CA	CLRL	-(SP)	
00000000G	8F	DD	001CC	PUSHL	#UAF\$ DEFERR	
65	03	FB	001D2	CALLS	#3, LIB\$SIGNAL	
0498	C3	B0	001D5	MOVW	UAFRAB+34, DEFAULT_SIZE	1184
0614	C3	9E	001DC	MOVAB	RECBUF, UAFRAB+36	1187
0618	C3	9E	001E2	MOVAB	RECBUF, UAFRAB+40	1188
F8	A3	D0	001E8	MOVL	#1, NETUAF_EXISTS	1196
		9F	001EC	PUSHAB	NAFFAB	1201
68	01	FB	001F0	CALLS	#1, SYSS\$OPEN	
63	50	D0	001F3	MOVL	R0, RMSERR	
20	50	E8	001F6	BLBS	R0, 13\$	



setup - open initial files

N 13

16-Sep-1984 02:16:54

14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 46

(4)

			50	63	D0	001F9	MOVL	RMSERR, R0	1206
	00018292		8F	50	D1	001FC	CMPL	R0, #98962	
				05	12	00203	BNEQ	12\$	
		F8		A3	D4	00205	CLRL	NETUAF_EXISTS	1208
				35	11	00208	BRB	15\$	
				50	DD	0020A	PUSHL	R0	1210
				7E	D4	0020C	CLRL	-(SP)	
				8F	DD	0020E	PUSHL	#UAF\$ NAONAF	
	65		00000000G	03	FB	00214	CALLS	#3, LIB\$SIGNAL	
				26	11	00217	BRB	15\$	1206
		0634		C3	9F	00219	PUSHAB	NAFRAB	1216
	6A			01	FB	0021D	CALLS	#1, SYS\$CONNECT	
	63			50	DD	00220	MOVL	R0, RMSERR	
	0B			50	EB	00223	BLBS	R0, 14\$	
			00000000G	8F	DD	00226	PUSHL	#UAF\$ NAFCONERR	1217
	65			01	FB	0022C	CALLS	#1, LIB\$SIGNAL	
				0E	11	0022F	BRB	15\$	
	0658	C3	058C	C3	9E	00231	MOVAB	NETBUF, NAFRAB+36	1223
	065C	C3	058C	C3	9E	00238	MOVAB	NETBUF, NAFRAB+40	1224
	03	A6	C0	8F	8A	0023F	BICB2	#192, IDENT+3	1231
02	A6			01	F0	00244	INSV	#1, #0, #14, IDENT+2	1232
		OE		04	B0	0024A	MOVW	#4, IDENT	1233
				7E	7C	0024D	CLRQ	-(SP)	1234
				7E	7C	0024F	CLRQ	-(SP)	
				7E	D4	00251	CLRL	-(SP)	
				66	DD	00253	PUSHL	IDENT	
	00000000G	00		06	FB	00255	CALLS	#6, SYS\$IDTOASC	
	00018292	8F		50	D1	0025C	CMPL	STATUS, #98962	1235
				04	12	00263	BNEQ	16\$	
				A3	D4	00265	CLRL	RDB_EXISTS	1236
					04	00268	RET		
	FC	A3		01	D0	00269	MOVL	#1, RDB_EXISTS	1237
				04	0026D	RET			1239

; Routine Size: 622 bytes, Routine Base: \$CODE\$ + 00FB

add\_uaf - insert new user record

```
1149 1240 1 %sbttl 'add_uaf - insert new user record'
1150 1241 1 global routine add_uaf : novalue =
1151 1242 2 begin
1152 1243 2
1153 1244 2 ++
1154 1245 2
1155 1246 2 FUNCTIONAL DESCRIPTION:
1156 1247 2
1157 1248 2 Routine to add new user to authorization file.
1158 1249 2
1159 1250 2 INPUTS:
1160 1251 2
1161 1252 2 none
1162 1253 2
1163 1254 2 IMPLICIT INPUTS:
1164 1255 2
1165 1256 2 none
1166 1257 2
1167 1258 2 OUTPUTS:
1168 1259 2
1169 1260 2 none
1170 1261 2
1171 1262 2 IMPLICIT OUTPUTS:
1172 1263 2
1173 1264 2 none
1174 1265 2
1175 1266 2 ROUTINE VALUE:
1176 1267 2
1177 1268 2 none
1178 1269 2
1179 1270 2 SIDE EFFECTS:
1180 1271 2
1181 1272 2 A record is added to SYSUAF.DAT
1182 1273 2 --
1183 1274 2
1184 1275 2 map
1185 1276 2 tokenptr : ref vector [,byte];
1186 1277 2
1187 1278 2 local
1188 1279 2 user_dsc : vector [2]; ! descriptor for username in record
1189 1280 2
1190 1281 2
1191 1282 2 Make sure a username was specified.
1192 1283 2
1193 1284 2
1194 1285 2 if not cli$present (sd_token1)
1195 1286 2 or not cli$get_value (sd_token1,tokendsc)
1196 1287 2 or .tokenlen eql 0
1197 1288 2 then return LIB$SIGNAL(UAF$_NOUSERNAME);
1198 1289 2
1199 1290 2
1200 1291 2 ADD must check that the username supplied is not too long.
1201 1292 2
1202 1293 2 ***if .tokenlen gtr uaf$_username
1203 1294 2 if .tokenlen gtr 12
1204 1295 2 then return LIB$SIGNAL(UAF$_NAMETOOBIG);
1205 1296 2
```

add\_uaf - insert new user record

```
1206 1297 2 |
1207 1298 2 | Make sure a legal username was entered, otherwise the account may not be
1208 1299 2 | accessible via LOGIN or the Input Symbiont.
1209 1300 2 |
1210 1301 2 | incru 1 to .tokenlen - 1
1211 1302 2 | do
1212 1303 2 |     if ch$fail (ch$find_ch (.symbol_str<0,8>,
1213 1304 2 |         symbol_str + 1,
1214 1305 2 |         .tokenptr [.i]))
1215 1306 2 |         then return LIB$SIGNAL(UAF$_INVUSERNAME);
1216 1307 2 | user_dsc[0] = .tokenlen;
1217 1308 2 | user_dsc[1] = recbuf[uaf$t_username];
1218 1309 2 |
1219 1310 2 |
1220 1311 2 | Move the default record to the current record buffer, so that
1221 1312 2 | fields which are not entered will receive the default
1222 1313 2 | value. Then insert the username just entered.
1223 1314 2 |
1224 1315 2 |
1225 1316 2 | ch$move (.default_size, default_record, recbuf);
1226 1317 2 | ch$copy (.tokenlen, .tokenptr, ' ', uaf$s_username, recbuf[uaf$t_username]);
1227 1318 2 |
1228 1319 2 |
1229 1320 2 | Call routine to fill in all supplied values. Exit if any errors
1230 1321 2 | were found.
1231 1322 2 |
1232 1323 2 |
1233 1324 2 | pwd_flag = true; ! plan to supply a password
1234 1325 2 | uaf$rab[ra$b$w_rs2] = .default_size;
1235 1326 2 | uaf$gl_ctlmsk[uaf$v_add] = true;
1236 1327 2 | if not update_record ()
1237 1328 2 | then
1238 1329 2 |     begin
1239 1330 2 |         uaf$gl_ctlmsk[uaf$v_add] = false;
1240 1331 2 |         return;
1241 1332 2 |     end;
1242 1333 2 |
1243 1334 2 | uaf$gl_ctlmsk[uaf$v_add] = false;
1244 1335 2 |
1245 1336 2 | if .pwd_flag ! if no explicit password
1246 1337 2 | then
1247 1338 2 |     begin
1248 1339 2 |         pwddsc[dsc$w_length] = .defpass<0,8>;
1249 1340 2 |         pwddsc[dsc$a_pointer] = defpass+1;
1250 1341 2 |         $gettim (timadr = time_buf);
1251 1342 2 |         recbuf[uaf$w_salt] = .time_buf<3*8,16>;
1252 1343 2 |         recbuf[uaf$b_encrypt] = encrypt;
1253 1344 2 |         lgi$hpwd (rec_encrypt dsc, pwddsc, .recbuf[uaf$b_encrypt],
1254 1345 2 |             .recbuf[uaf$w_salt], user_dsc);
1255 1346 2 |     end;
1256 1347 2 |
1257 1348 2 |
1258 1349 2 | Now output the new record.
1259 1350 2 |
1260 1351 2 |
1261 1352 2 | if rmsbad ($put (rab = uaf$rab))
1262 1353 2 | then
```

add\_uaf - insert new user record

```
1263 1354 2 if .rmserr eql rms$ dup
1264 1355 2 then return LIB$SIGNAL(UAF$_UAEERR)
1265 1356 2 else LIB$SIGNAL(UAF$_ADDERR, 0, .rmserr)
1266 1357 2 else
1267 1358 2 begin
1268 1359 2
1269 1360 2
1270 1361 3 Tell user that addition was successful. Note that file was changed.
1271 1362 3
1272 1363 3 LIB$SIGNAL(UAF$_ADDMSG);
1273 1364 4 if (.uaf$gl_ctlmsk[uaf$y_cli]
1274 1365 4 and (not .uaf$gl_ctlmsk[uaf$y_clitables]))
1275 1366 4 then LIB$SIGNAL(UAF$_CLIWARNMSG);
1276 1367 4 security_audit (nsa$recid_sysuaf_add);
1277 1368 3 modify iflag = true;
1278 1369 4 if (cli$present (sd_add_identifier) and
1279 1370 4 .rdb_exists )
1280 1371 3 then
1281 1372 4 begin
1282 1373 4
1283 1374 4 Add the appropriate identifiers.
1284 1375 4 Set the resource attribute if /ATTRIB=RESOURCE was specified
1285 1376 4 and then add the appropriate identifiers to the rights data base
1286 1377 4
1287 1378 4 if cli$present (sd_attribresource)
1288 1379 4 then attributes = kgb$m_resource
1289 1380 4 else attributes = 0 ;
1290 1381 4 uaf$add_ident_recbuf ( ) ;
1291 1382 3 end ;
1292 1383 2 end;
1293 1384 1 end;
```

```
OFFC 00000
SB 00000000G 00 9E 00002
SA 00000000' 00 9E 00009
S9 00000000' 00 9E 00010
S8 00000000' 00 9E 00017
SE 08 C2 0001E
59 DD 00021
00000000G 00 01 FB 00023
12 50 E9 0002A
58 DD 0002D
59 DD 0002F
00000000G 00 02 FB 00031
04 50 E9 00038
68 B5 0003B
08 12 0003D
0C000000G 8F DD 0003F 1$:
34 11 00045
0C 68 B1 00047 2$:
08 1B 0004A
00000000G 8F DD 0004C
```

.EXTRN SYSS\$GETTIM

```
.ENTRY ADD_UAF, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- 1241
R11
MOVAB LIB$SIGNAL, R11
MOVAB PWDDSC, R10
MOVAB SD_TOKEN1, R9
MOVAB TORENDSC, R8
SUBL2 #8, SP
PUSHL R9 1285
CALLS #1, CLI$PRESENT
BLBC R0, 1$
PUSHL R8 1286
PUSHL R9
CALLS #2, CLI$GET_VALUE
BLBC R0, 1$
TSTW TOKENLEN 1287
BNEQ 2$
PUSHL #UAF$_NOUSERNAME 1288
BRB 6$
CMPW TOKENLEN, #12 1294
BLEQU 3$
PUSHL #UAF$_NAMETOOBIG 1295
```



			27	11	00052	BRB	6\$		
		53	68	3C	00054	3\$: MOVZWL	TOKENLEN, R3		1301
			53	D7	00057	DECL	R3		
			52	D4	00059	CLRL	I		
			23	11	0005B	BRB	8\$		
		51	C9	9A	0005D	4\$: MOVZBL	SYMBOL_STR, R1		1303
		50	A8	D0	00062	MOVL	TOKENPTR, R0		1305
01ED	C9	51	6240	3A	00066	LOCC	(I)[R0], R1, SYMBOL_STR+1		
			02	12	0006D	BNEQ	5\$		
			51	D4	0006F	CLRL	R1		
			51	D5	00071	5\$: TSTL	R1		
			09	12	00073	BNEQ	7\$		
		00000000G	8F	DD	00075	PUSHL	#UAF\$_INVUSERNAME		1306
			00AF	31	0007B	6\$: BRW	11\$		
			52	D6	0007E	7\$: INCL	I		1305
		53	52	D1	00080	8\$: CMPL	I, R3		
			D8	1B	00083	BLEQU	4\$		
		57	68	3C	00085	MOVZWL	TOKENLEN, R7		1307
		6E	57	D0	00088	MOVL	R7, USER_DSC		
		04	AE	9E	0008B	MOVAB	RECBUF+4, USER_DSC+4		1308
		56	FA78	CA	3C	00090	MOVZWL	DEFAULT_SIZE, R6	1316
20	A8	FA7C	56	28	00095	MOVCS	R6, DEFAULT_RECORD, RECBUF		
			50	04	AB	D0	0009C	MOVL	TOKENPTR, R0
20	20	60	57	2C	000A0	MOVCS	R7, (R0), #32, #32, RECBUF+4		1317
			24	A8	000A5				
		0698	C8	01	D0	000A7	MOVL	#1, PWD_FLAG	1324
		062A	C8	56	B0	000AC	MOVW	R6, UAFRAB+34	1325
		A4	A8	02	88	000B1	BISB2	#2, UAF\$GL_CTLMSK	1326
		00000000G	00	00	FB	000B5	CALLS	#0, UPDATE_RECORD	1327
			05	50	E8	000BC	BLBS	R0, 9\$	
		A4	A8	02	8A	000BF	BICB2	#2, UAF\$GL_CTLMSK	1330
				04	000C3	RET			1329
		A4	A8	02	8A	000C4	9\$: BICB2	#2, UAF\$GL_CTLMSK	1334
		3B	0698	C8	E9	000C8	BLBC	PWD_FLAG, T0\$	1336
		6A	0120	C9	9B	000CD	MOVZBW	DEFPASS, PWDDSC	1339
		04	AA	0121	C9	9E	000D2	MOVAB	DEFPASS+1, PWDDSC+4
			0690	C8	9F	000D8	PUSHAB	TIME_BUF	1341
		00000000G	00	01	FB	000DC	CALLS	#1, SYSSGETTIM	
		0186	C8	0693	C8	B0	000E3	MOVW	TIME_BUF+3, RECBUF+358
		0188	C8	02	90	000EA	MOVB	#2, RECBUF+360	1343
			7E	5E	DD	000EF	PUSHL	SP	1344
			7E	0186	C8	3C	000F1	MOVZWL	RECBUF+358, -(SP)
				0188	C8	9A	000F6	MOVZBL	RECBUF+360, -(SP)
					5A	DD	000FB	PUSHL	R10
				01E4	C9	9F	000FD	PUSHAB	REC_ENCRYPT_DSC
		00000000G	00	05	FB	00101	CALLS	#5, LGISHPWD	
		00000000G	00	0608	C8	9F	00108	10\$: PUSHAB	UAFRAB
		18	A8	01	FB	0010C	CALLS	#1, SYSSPUT	1352
			25	50	D0	00113	MOVL	R0, RMSERR	
			50	E8	00117	BLBS	R0, 13\$		
			8F	18	A8	D0	0011A	MOVL	RMSERR, R0
000184EC			50	D1	0011E	CMPL	R0, #99564		1354
			0A	12	00125	BNEQ	12\$		
		00000000G	8F	DD	00127	PUSHL	#UAF\$ UAEERR		1355
		6B	01	FB	0012D	11\$: CALLS	#1, LTBSSIGNAL		
			04	00130	RET				
			50	DD	00131	12\$: PUSHL	R0		1356

add\_uaf - insert new user record

F 14  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1Page 51  
(5)

			7E	D4	00133	CLRL	-(SP)	
		00000000G	8F	DD	00135	PUSHL	#UAF\$_ADDERR	
		6B	03	FB	0013B	CALLS	#3, LTB\$SIGNAL	
				04	0013E	RET		1354
		00000000G	8F	DD	0013F	PUSHL	#UAF\$_ADDMSG	1363
		6B	01	FB	00145	CALLS	#1, LTB\$SIGNAL	
0E	A4	AB	03	E1	0014B	BBC	#3, UAF\$GL_CTLMSK, 14\$	1364
09	A4	AB	04	E0	0014D	BBS	#4, UAF\$GL_CTLMSK, 14\$	1365
		00000000G	8F	DD	00152	PUSHL	#UAF\$_CLIWARNMSG	1366
		6B	01	FB	00158	CALLS	#1, LTB\$SIGNAL	
		00010002	8F	DD	0015B	PUSHL	#65538	1367
			01	FB	00161	CALLS	#1, SECURITY AUDIT	
00000000V	00		01	D0	00168	MOVL	#1, MODIFY FLAG	1368
F624	CA			A9	9F	PUSHAB	SD_ADD_IDENTIFIER	1369
		44	01	FB	00170	CALLS	#1, CLISPRESENT	
00000000G	00		50	E9	00177	BLBC	R0, 17\$	
	28		A8	E9	0017A	BLBC	RDB EXISTS, 17\$	1370
	24	14	CA	9F	0017E	PUSHAB	SD_ATTRIBUTESOURCE	1378
		F5DB	01	FB	00182	CALLS	#1, CLISPRESENT	
00000000G	00		50	E9	00189	BLBC	R0, 15\$	
	09		01	D0	0018C	MOVL	#1, ATTRIBUTES	1379
00000000G	00		06	11	00193	BRB	16\$	
		00000000G	00	D4	00195	CLRL	ATTRIBUTES	1380
			00	FB	0019B	CALLS	#0, UAF\$ADD_IDENT_RECBUF	1381
			04	001A2	17\$:	RET		1384

; Routine Size: 419 bytes, Routine Base: \$CODE\$ + 0369

## add\_proxy - insert new proxy record

```
1295 1385 1 %sbttl 'add_proxy - insert new proxy record'
1296 1386 1 global routine add_proxy : novalue =
1297 1387 2 begin
1298 1388 2
1299 1389 2 ++
1300 1390 2
1301 1391 2 FUNCTIONAL DESCRIPTION:
1302 1392 2
1303 1393 2 This routine adds an entry to the NETUAF.DAT Proxy Login File
1304 1394 2
1305 1395 2 INPUTS:
1306 1396 2
1307 1397 2 none
1308 1398 2
1309 1399 2 OUTPUTS:
1310 1400 2
1311 1401 2 none
1312 1402 2
1313 1403 2 IMPLICIT INPUTS:
1314 1404 2
1315 1405 2 TOKENLEN, TOKENPTR
1316 1406 2
1317 1407 2 IMPLICIT OUTPUTS:
1318 1408 2
1319 1409 2 none
1320 1410 2
1321 1411 2 ROUTINE VALUE:
1322 1412 2
1323 1413 2 none
1324 1414 2
1325 1415 2 SIDE EFFECTS:
1326 1416 2
1327 1417 2 A record is added to NETUAF.DAT
1328 1418 2
1329 1419 2 --
1330 1420 2
1331 1421 2 local
1332 1422 2 node_len,
1333 1423 2 node_ptr,
1334 1424 2 remuser_len,
1335 1425 2 remuser_ptr,
1336 1426 2 locuser_len,
1337 1427 2 locuser_ptr;
1338 1428 2
1339 1429 2
1340 1430 2 Can't do anything if there is no NETUAF.DAT...
1341 1431 2
1342 1432 2 if not .netuaf exists
1343 1433 2 then return LIB$SIGNAL(UAF$_NAFDNE);
1344 1434 2
1345 1435 2
1346 1436 2 Clear NETUAF.DAT buffer
1347 1437 2
1348 1438 2 ch$fill (' ', naf$c_length, netbuf);
1349 1439 2
1350 1440 2
1351 1441 2 Retrieve token from command line
```

add\_proxy - insert new proxy record

```
1352 1442 2 1
1353 1443 2 2 cli$get_value (sd_token1, tokendsc);
1354 1444 2 3
1355 1445 2 4
1356 1446 2 5 Make sure entry is in proper node::remoteuser format
1357 1447 2 6
1358 1448 2 7 if not remote_parse (node_ptr, node_len, remuser_ptr, remuser_len)
1359 1449 2 8 then return;
1360 1450 2 9
1361 1451 2 10 Fill in NETBUF with new remotename field
1362 1452 2 11
1363 1453 2 12 ch$copy (.node_len, .node_ptr, ' ', naf$s_node, netbuf[naf$t_node]);
1364 1454 2 13 ch$copy (.remuser_len, .remuser_ptr, ' ', naf$s_remuser, netbuf[naf$t_remuser]);
1365 1455 2 14
1366 1456 2 15
1367 1457 2 16 Now get second token, the local user name
1368 1458 2 17
1369 1459 2 18 cli$get_value (sd_token2, tokendsc);
1370 1460 2 19
1371 1461 2 20 locuser_len = .tokenlen;
1372 1462 2 21 locuser_ptr = .tokenptr;
1373 1463 2 22 ***if .locuser_len gtru naf$s_localuser
1374 1464 2 23 if .locuser_len gtru 12
1375 1465 2 24 then return LIB$SIGNAL(UAF$_NAMETOOBIG);
1376 1466 2 25
1377 1467 2 26
1378 1468 2 27 If local name is *, then use same name as remote user
1379 1469 2 28
1380 1470 2 29 if .tokenlen eql 1 and .(.tokenptr)<0,8> eql '*'
1381 1471 2 30 then
1382 1472 2 31 begin
1383 1473 2 32 locuser_len = .remuser_len;
1384 1474 2 33 ch$move (naf$s_remuser, netbuf[naf$t_remuser], netbuf[naf$t_localuser]);
1385 1475 2 34 end
1386 1476 2 35
1387 1477 2 36 Otherwise just copy into localuser field in NETBUF
1388 1478 2 37
1389 1479 2 38 else
1390 1480 2 39 ch$copy (.locuser_len, .locuser_ptr, ' ',
1391 1481 2 40 naf$s_localuser, netbuf[naf$t_localuser]);
1392 1482 2 41
1393 1483 2 42
1394 1484 2 43 Make sure that the local user does indeed exist in SYSUAF.DAT
1395 1485 2 44 (unless local user is *)
1396 1486 2 45
1397 1487 2 46 if not locate_user (.locuser_len, netbuf[naf$t_localuser], 0)
1398 1488 2 47 and not (.locuser_len eql 1 and .(netbuf[naf$t_localuser])<0,8> eql '*')
1399 1489 2 48 then return LIB$SIGNAL(UAF$_BADUSR, 2, .locuser_len, netbuf[naf$t_localuser]);
1400 1490 2 49
1401 1491 2 50 nafrab[rab$w_rsz] = naf$c_length;
1402 1492 2 51
1403 1493 2 52
1404 1494 2 53 Add NETUAF.DAT record
1405 1495 2 54
1406 1496 2 55 if rmsbad ($put (rab = nafrab))
1407 1497 2 56 then
1408 1498 2 57 begin
```



add\_proxy - insert new proxy record

```
1409      if .rmserr eql rms$_dup
1410      then
1411          return LIB$SIGNAL(UAF$_NAFUAEERR)
1412      else
1413          LIB$SIGNAL(UAF$_NAFADDERR, 0, .rmserr)
1414      end
1415  else
1416      begin
1417          LIB$SIGNAL(UAF$_NAFADDMSG);
1418          security_audit (nsa$_recid_netuaf_add);
1419      end;
1420  netuaf_modified = true;
1421  end;
1422
```

				07FC 00000	.ENTRY	ADD PROXY, Save R2,R3,R4,R5,R6,R7,R8,R9,R10	1386
		5A	00000000G	00 9E 00002	MOVAB	CLISGET VALUE, R10	
		59	00000000G	00 9E 00009	MOVAB	LIB\$SIGNAL, R9	
		58	00000000'	00 9E 00010	MOVAB	NETBUF+64, R8	
		5E		10 C2 00017	SUBL2	#16, SP	
		08	FA2C	C8 E8 0001A	BLBS	NETUAF EXISTS, 1\$	1432
			00000000G	8F DD 0001F	PUSHL	#UAF\$_NAFDNE	1433
				5F 11 00025	BRB	3\$	
0064	8F	20		00 2C 00027 1\$:	MOVCS	#0, (SP), #32, #100, NETBUF	1438
			CO	A8 0002E			
			FA1C	C8 9F 00030	PUSHAB	TOKENDSC	
			00000000'	00 9F 00034	PUSHAB	SD_TOKEN1	1443
		6A		02 FB 0003A	CALLS	#2, CLISGET_VALUE	
				5E DD 0003D	PUSHL	SP	1448
			08	AE 9F 0003F	PUSHAB	REMUSER_PTR	
			10	AE 9F 00042	PUSHAB	NODE_LEN	
			18	AE 9F 00045	PUSHAB	NODE_PTR	
		00000000V	00	04 FB 00048	CALLS	#4, REMOTE_PARSE	
			01	50 E8 0004F	BLBS	R0, 2\$	
				04 00052	RET		
20	20	0C	BE	AE 2C 00053 2\$:	MOVCS	NODE_LEN, @NODE_PTR, #32, #32, NETBUF	1453
			CO	A8 0005A			
20	20	04	BE	6E 2C 0005C	MOVCS	REMUSER_LEN, @REMUSER_PTR, #32, #32, -	1454
			E0	A8 00062		NETBUF+32	
			FA1C	C8 9F 00064	PUSHAB	TOKENDSC	1459
			00000000'	00 9F 00068	PUSHAB	SD_TOKEN2	
		6A		02 FB 0006E	CALLS	#2, CLISGET_VALUE	
		56	FA1C	C8 3C 00071	MOVZWL	TOKENLEN, LOCUSER_LEN	1461
		57	FA20	C8 D0 00076	MOVL	TOKENPTR, LOCUSER_PTR	1462
		0C		56 D1 0007B	CMPL	LOCUSER_LEN, #12	1464
				08 1B 0007E	BLEQU	4\$	
			00000000G	8F DD 00080	PUSHL	#UAF\$_NAMETOOBIG	1465
				77 11 00086 3\$:	BRB	9\$	
		01	FA1C	C8 B1 00088 4\$:	CMPL	TOKENLEN, #1	1470
				14 12 0008D	BNEQ	5\$	
		50	FA20	C8 D0 0008F	MOVL	TOKENPTR, R0	
		2A		60 91 00094	CMPB	(R0), #42	
				0A 12 00097	BNEQ	5\$	

## add\_proxy - insert new proxy record

J 14  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1Page 55  
(6)

20	68	E0	56 A8	6E 20	D0 28	00099 0009C	MOVL MOVCS	REMUSER LEN, LOCUSER LEN #32, NETBUF+32, NETBOF+64	1473 1474
				06	11	000A1	BRB	6\$	1470
	20		67	56	2C	000A3	5\$: MOVCS	LOCUSER_LEN, (LOCUSER_PTR), #32, #32, - NETBUF+64	1481
				68		000A8			
				7E	D4	000A9	6\$: CLRL	-(SP)	1487
		0140		8F	BB	000AB	PUSHR	#M<R6,R8>	
		00000000V	00	03	FB	000AF	CALLS	#3, LOCATE_USER	
			1A	50	E8	000B6	BLBS	R0, 8\$	
			01	56	D1	000B9	CMPL	LOCUSER_LEN, #1	1488
				05	12	000BC	BNEQ	7\$	
			2A	68	91	000BE	CMPB	NETBUF+64, #42	
				10	13	000C1	BEQL	8\$	
				8F	BB	000C3	7\$: PUSHR	#M<R6,R8>	1489
				02	DD	000C7	PUSHL	#2	
				8F	DD	000C9	PUSHL	#UAF\$BADUSR	
			69	04	FB	000CF	CALLS	#4, LIB\$SIGNAL	
				04		000D2	RET		
		008A	C8	8F	9B	000D3	8\$: MOVZBW	#100, NAFRAB+34	1491
				A8	9F	000D9	PUSHAB	NAFRAB	1496
		00000000G	00	01	FB	000DC	CALLS	#1, SY\$SPUT	
		FA34	C8	50	D0	000E3	MOVL	R0, RMSERR	
			27	50	E8	000E8	BLBS	R0, 11\$	
			50	C8	D0	000EB	MOVL	RMSERR, R0	1499
		000184EC	8F	50	D1	000F0	CMPL	R0, #99564	
				0A	12	000F7	BNEQ	10\$	
				8F	DD	000F9	PUSHL	#UAF\$NAFUAERR	1501
			69	01	FB	000FF	9\$: CALLS	#1, LIB\$SIGNAL	
				04		00102	RET		
				50	DD	00103	10\$: PUSHL	R0	1503
				7E	D4	00105	CLRL	-(SP)	
				8F	DD	00107	PUSHL	#UAF\$NAFADDERR	
			69	03	FB	0010D	CALLS	#3, LIB\$SIGNAL	
				16	11	00110	BRB	12\$	1498
				8F	DD	00112	11\$: PUSHL	#UAF\$NAFADDMSG	1507
			69	01	FB	00118	CALLS	#1, LIB\$SIGNAL	
				8F	DD	0011B	PUSHL	#65539	1508
				01	FB	00121	CALLS	#1, SECURITY_AUDIT	
		00000000V	00	01	D0	00128	12\$: MOVL	#1, NETUAF_MODIFIED	1511
		00000000'	00	04		0012F	RET		1512

; Routine Size: 304 bytes, Routine Base: \$CODE\$ + 050C

remote\_parse - parses 'node::remoteuser'

```
1424 1513 1 %sbttl 'remote_parse - parses 'node::remoteuser''
1425 1514 1 routine remote_parse (node_ptr, node_len, remuser_ptr, remuser_len) =
1426 1515 2 begin
1427 1516 2
1428 1517 2 ++
1429 1518 2
1430 1519 2 FUNCTIONAL DESCRIPTION:
1431 1520 2
1432 1521 2 This routine parses a remote user specification in the form
1433 1522 2 node::remuser, and returns the two components by lengths
1434 1523 2 and pointers to the strings
1435 1524 2
1436 1525 2 INPUTS:
1437 1526 2
1438 1527 2 node_ptr - returned as pointer to nodename
1439 1528 2 node_len - length of nodename
1440 1529 2 remuser_ptr - returned as pointer to remote user name
1441 1530 2 remuser_len - length of remote user name
1442 1531 2
1443 1532 2 IMPLICIT INPUTS:
1444 1533 2
1445 1534 2 TOKENLEN and TOKENPTR - the remote user specification is assumed
1446 1535 2 to have just been fetched from the command line
1447 1536 2
1448 1537 2 OUTPUTS:
1449 1538 2
1450 1539 2 none
1451 1540 2
1452 1541 2 ROUTINE VALUE:
1453 1542 2
1454 1543 2 TRUE if parsed successfully
1455 1544 2 FALSE if error encountered in parsing
1456 1545 2
1457 1546 2 --
1458 1547 2
1459 1548 2 map
1460 1549 2 dbl_colon : vector;
1461 1550 2
1462 1551 2 local
1463 1552 2 dbl_colon_ptr;
1464 1553 2
1465 1554 2
1466 1555 2 Better be able to find a double colon in the remotename...
1467 1556 2
1468 1557 2 dbl_colon_ptr = ch$find_sub (.tokenlen, .tokenptr, 2, .dbl_colon [1]);
1469 1558 2
1470 1559 2 if .dbl_colon_ptr eql 0 ! no double colon found
1471 1560 2 or .dbl_colon_ptr eql .tokenptr ! no node found
1472 1561 2 or .dbl_colon_ptr eql (.tokenptr + .tokenlen - 2) ! no remote user found
1473 1562 2 then return LIB$SIGNAL(UAF$_BADNODFORM);
1474 1563 2
1475 1564 2
1476 1565 2 Determine node length and pointer
1477 1566 2
1478 1567 2 .node_len = .dbl_colon_ptr - .tokenptr;
1479 1568 2 .node_ptr = .tokenptr;
1480 1569 2
```

remote\_parse - parses 'node::remoteuser'

```
1481 1570 2 |
1482 1571 2 | Make sure node name isn't too long
1483 1572 2 |
1484 1573 2 | ***if . (.node_len) gtru naf$s_node
1485 1574 2 | if . (.node_len) gtru 6
1486 1575 2 | then return LIB$SIGNAL(UAF$_NODTOOBIG);
1487 1576 2 |
1488 1577 2 |
1489 1578 2 | Determine remote username length and pointer
1490 1579 2 |
1491 1580 2 | .remuser_len = .tokenlen - .(.node_len) - 2;
1492 1581 2 | .remuser_ptr = .dbl_colon_ptr + 2;
1493 1582 2 |
1494 1583 2 |
1495 1584 2 | And make sure name isn't too long
1496 1585 2 |
1497 1586 2 | ***if . (.remuser_len) gtru naf$s_remuser
1498 1587 2 | if . (.remuser_len) gtru 12
1499 1588 2 | then return LIB$SIGNAL(UAF$_NAMETOOBIG);
1500 1589 2 |
1501 1590 2 | return true;
1502 1591 1 | end;
```

				007C 00000 REMOTE_PARSE:				
				56 00000000'	00 9E 00002	WORD	Save R2,R3,R4,R5,R6	1514
				55	66 3C 00009	MOVAB	TOKENLEN, R6	
				54 04	A6 D0 0000C	MOVZWL	TOKENLEN, R5	1557
				50 00000000'	00 D0 00010	MOVL	TOKENPTR, R4	
64				60	02 39 00017	MOVL	DBL_COLON+4, R0	
					03 13 0001C	MATCHC	#2, (R0), R5, (R4)	
				53	02 D0 0001E	BEQL	1\$	
				53	02 C2 00021 1\$:	MOVL	#2, R3	
					0F 13 00024	SUBL2	#2, R3	
				54	53 D1 00026	BEQL	2\$	1559
					0A 13 00029	CMPL	DBL_COLON_PTR, R4	1560
				50 FE A544	9E 0002B	BEQL	2\$	
				50	53 D1 00030	MOVAB	-2(R5)[R4], R0	1561
					08 12 00033	CMPL	DBL_COLON_PTR, R0	
					8F DD 00035 2\$:	BNEQ	3\$	
					38 11 0003B	PUSHL	#UAF\$_BADNODFORM	1562
				50 04	A6 D0 0003D 3\$:	BRB	5\$	
				53	50 C3 00041	MOVL	TOKENPTR, R0	1567
08	BC			BC	50 D0 00046	SUBL3	R0, DBL_COLON_PTR, @NODE_LEN	
		04		06 08	BC D1 0004A	MOVL	R0, @NODE_PTR	1568
					08 1B 0004E	CMPL	@NODE_LEN, #6	1574
					8F DD 00050	BLEQU	4\$	
					1D 11 00056	PUSHL	#UAF\$_NODTOOBIG	1575
				50	66 3C 00058 4\$:	BRB	5\$	
				50	08 BC C2 0005B	MOVZWL	TOKENLEN, R0	1580
				10 BC FE A0	9E 0005F	SUBL2	@NODE_LEN, R0	
				0C BC	02 A3 9E 00064	MOVAB	-2(R0), @REMUSER_LEN	
				0C	10 BC D1 00069	MOVAB	2(R3), @REMUSER_PTR	1581
						CMPL	@REMUSER_LEN, #T2	1587



UAFMAIN  
V04-000

remote\_parse - parses 'node::remoteuser'

M 14  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 58  
(7)

		0E	1B	0006D	BLEQU	6\$	:	
		8F	DD	0006F	PUSHL	#UAF\$ NAMETOOBIG	:	1588
00000000G	00	01	FB	00075	CALLS	#1, LIB\$SIGNAL	:	
			04	0007C	RET		:	
	50	01	D0	0007D	MOVL	#1, R0	:	1590
			04	00080	RET		:	1591

; Routine Size: 129 bytes,      Routine Base: \$CODE\$ + 063C

UA  
VO

copy\_uaf - copy user record

```
1504 1592 1 %sbttl 'copy_uaf - copy user record'
1505 1593 1 global routine copy_uaf =
1506 1594 3 begin
1507 1595 3
1508 1596 3 ++
1509 1597 3
1510 1598 3 FUNCTIONAL DESCRIPTION:
1511 1599 3
1512 1600 3 Routine to copy a user authorization record, giving the
1513 1601 3 new authorization record a different name.
1514 1602 3
1515 1603 3 INPUTS:
1516 1604 3
1517 1605 3 none
1518 1606 3
1519 1607 3 IMPLICIT INPUTS:
1520 1608 3
1521 1609 3 none
1522 1610 3
1523 1611 3 OUTPUTS:
1524 1612 3
1525 1613 3 none
1526 1614 3
1527 1615 3 ROUTINE VALUE:
1528 1616 3
1529 1617 3 false if the copy fails;
1530 1618 3 true if the copy succeeds.
1531 1619 3
1532 1620 3 SIDE EFFECTS:
1533 1621 3
1534 1622 3 A user record is added.
1535 1623 3
1536 1624 3 --
1537 1625 3
1538 1626 3 local
1539 1627 3
1540 1628 3 status,
1541 1629 3 flag,
1542 1630 3 lock_rec,
1543 1631 3 def_sys,
1544 1632 3 old_user_buffer : vector [uaf$s_username, byte];
1545 1633 3
1546 1634 3 map
1547 1635 3 tokenptr : ref vector [,byte];
1548 1636 3
1549 1637 3 uaf$gl_ctlmsk[uaf$v_copy] = not .uaf$gl_ctlmsk[uaf$v_rename];
1550 1638 3
1551 1639 3 If this is a COPY directly from the UAF> prompt, the authorization
1552 1640 3 record need not be locked, and the default and system records may be
1553 1641 3 copied. HOWEVER, if this COPY is part of a RENAME operation, the record
1554 1642 3 must be locked, and the default and system records may not be renamed.
1555 1643 3 (The RENAME operation is similar to the COPY operation except that
1556 1644 3 the original record is REMOVE'd. COPY leaves both records.)
1557 1645 3
1558 1646 3 if not .uaf$gl_ctlmsk[uaf$v_rename]
1559 1647 3 then
1560 1648 3 begin
```

copy\_uaf - copy user record

```
1561 1649 lock_rec = false;
1562 1650 def_sys = true;
1563 1651 flag = false;
1564 1652 end
1565 1653 else
1566 1654 begin
1567 1655 lock_rec = true;
1568 1656 def_sys = false;
1569 1657 flag = true;
1570 1658 end;
1571 1659
1572 1660
1573 1661 Place record to be copied into RECBUF
1574 1662 (If the third argument is true, the call to GET USER RECORD
1575 1663 is part of a RENAME operation, and the first token should be saved.
1576 1664 If the third argument is false, the call is part of a COPY
1577 1665 operation, and the first token need not be saved.)
1578 1666
1579 1667
1580 1668 if get_user_record (.lock_rec, .def_sys, .flag)
1581 1669 then
1582 1670 begin
1583 1671 if not cli$present (sd_token2)
1584 1672 or not cli$get_value (sd_token2, tokendsc)
1585 1673 or .tokenlen eql 0
1586 1674 then return LIB$SIGNAL(UAF$_NOUSERNAME);
1587 1675
1588 1676
1589 1677 Make sure that the new username isn't too long
1590 1678
1591 1679 *** if .tokenlen gtru uaf$_username
1592 1680 if .tokenlen gtru 12
1593 1681 then LIB$SIGNAL(UAF$_NAMETOOBIG);
1594 1682
1595 1683
1596 1684 Make sure that the new username is legal
1597 1685
1598 1686 incru i to .tokenlen - 1
1599 1687 do
1600 1688 if ch$fail (ch$find_ch (.symbol_str<0,8>,
1601 1689 symbol_str + 1,
1602 1690 .tokenptr [.i]))
1603 1691 then return LIB$SIGNAL(UAF$_INVUSERNAME);
1604 1692
1605 1693
1606 1694 If this is a rename save the new user name
1607 1695
1608 1696 if .uaf$gl_ctlmsk[uaf$_v_rename]
1609 1697 then
1610 1698 begin
1611 1699 ch$move (.tokenlen, .tokenptr, newusername);
1612 1700 newuserlen = .tokenlen;
1613 1701 end;
1614 1702
1615 1703
1616 1704 Place the new username in RECBUF, but save the old username first
1617 1705
```

copy\_uaf - copy user record

```
1618 1706 3      ch$move (uaf$s_username, recbuf[uaf$t_username], old_user_buffer);
1619 1707 3      ch$copy (.tokenlen, .tokenptr,
1620 1708 3          uaf$s_username, recbuf[uaf$t_username]);
1621 1709 3      pwd_flag = true;
1622 1710 3
1623 1711 3      status = update_record ();
1624 1712 3      if not .status
1625 1713 3      then return false;
1626 1714 3
1627 1715 3      ---
1628 1716 3      If this is a copy operation then zero fill the last login info
1629 1717 3
1630 1718 3      if not .uaf$gl_ctlmsk[uaf$v_rename]
1631 1719 3      then
1632 1720 3          begin
1633 1721 3              recbuf[uaf$w_logfails] = 0 ;
1634 1722 3              ch$fill ( 0, uaf$s_lastlogin_i, recbuf[uaf$q_lastlogin_i] ) ;
1635 1723 3              ch$fill ( 0, uaf$s_lastlogin_n, recbuf[uaf$q_lastlogin_n] ) ;
1636 1724 3              end ;
1637 1725 3
1638 1726 3      ---
1639 1727 3      Now output the new record
1640 1728 3
1641 1729 3      if rmsbad ($put (rab = uaf$rab))
1642 1730 3      then
1643 1731 3          begin
1644 1732 3              if .rmserr eql rms$_dup
1645 1733 3              then
1646 1734 3                  return LIB$SIGNAL(UAF$_UAEERR)      ! username already exists
1647 1735 3              else
1648 1736 3                  begin
1649 1737 3                      LIB$SIGNAL(UAF$_ADDERR, 0, .rmserr);
1650 1738 3                      return false;
1651 1739 3                  end
1652 1740 3              end
1653 1741 3          else
1654 1742 3      ---
1655 1743 3      The copy was successful -- tell the user and set modify flag
1656 1744 3
1657 1745 3      begin
1658 1746 3          modify flag = true;
1659 1747 3          security_audit ((if not .uaf$gl_ctlmsk[uaf$v_rename]
1660 1748 3                          then nsak_recid_sysuaf_cop
1661 1749 3                          else nsak_recid_sysuaf_ren),
1662 1750 3                          old_user_buffer);
1663 1751 3          if not .uaf$gl_ctlmsk[uaf$v_rename]
1664 1752 3          then
1665 1753 3              begin
1666 1754 3                  LIB$SIGNAL(UAF$_COPMSG);
1667 1755 3                  if (.uaf$gl_ctlmsk[uaf$v_cli]
1668 1756 3                      and not .uaf$gl_ctlmsk[uaf$v_clitables])
1669 1757 3                      and .uaf$gl_ctlmsk[uaf$v_clitab_pres]
1670 1758 3                      then LIB$SIGNAL(UAF$_CLIQARNMSG);
1671 1759 3                  ---
1672 1760 3                  Since passwords are folded in with the username, passwords for
1673 1761 3                  COPIed records will no longer work--warn the user
1674 1762 3                  ---
```



copy\_uaf - copy user record

```
1675 1763 5      if .pwd_flag
1676 1764 5      then LIB$SIGNAL(UAF$DEFPWD);
1677 1765 5      uaf$gl_ctlmsk[uaf$u_copy] = false;
1678 1766 6      if (clt$present ( sd_add_identifier ) and
1679 1767 6          .rdb_exists )
1680 1768 5      then
1681 1769 6          begin
1682 1770 6              Set the resource attribute if /ATTRIB=RESOURCE was specified
1683 1771 6              and then add the appropriate identifiers to the rights data base
1684 1772 6              if clt$present (sd_attribresource)
1685 1773 6                  then attributes = kgb$m_resource
1686 1774 6                  else attributes = 0 ;
1687 1775 6              uaf$add_ident_recbuf ( ) ;
1688 1776 6          end;
1689 1777 6      end;
1690 1778 5      end;
1691 1779 4      end;
1692 1780 3      end;
1693 1781 2      end;
1694 1782 1      The attempt to GET_USER_RECORD failed...
1695 1783 1      else return false;
1696 1784 1      If we get here, everything succeeded -- return true
1697 1785 1      return true;
1698 1786 1      return true;
1699 1787 1      return true;
1700 1788 1      return true;
1701 1789 1      return true;
1702 1790 1      return true;
1703 1791 1      return true;
1704 1792 1      return true;
1705 1793 1      end;
```

			OFFC 00000	.ENTRY	COPY_UAF. Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; R11	
		5B 00000000G	00 9E 00002	MOVAB	CLISPRESENT, R11	
		5A 00000000'	00 9E 00009	MOVAB	NEWUSERNAME, R10	
		59 00000000'	00 9E 00010	MOVAB	SD TOKEN2, R9	
		58 00000000G	00 9E 00017	MOVAB	LIB\$SIGNAL, R8	
		57 00000000'	00 9E 0001E	MOVAB	UAF\$GL_CTLMSK, R7	
		5E	20 C2 00025	SUBL2	#32, SP	
50	67	01	01 EF 00028	EXTZV	#0, #1, UAF\$GL_CTLMSK, R0	1637
		50	50 92 0002D	MCOMB	R0, R0	
67	01	02	50 F0 00030	INSV	R0, #2, #1, UAF\$GL_CTLMSK	
		07	67 E8 00035	BLBS	UAF\$GL_CTLMSK, 1\$	1646
		51	01 7D 00038	MOVQ	#1, DEF_SYS	1650
			50 D4 0003B	CLRL	FLAG	1651
			06 11 0003D	BRB	2\$	1646
		52	01 D0 0003F 1\$:	MOVL	#1, LOCK_REC	1655
		50	01 7D 00042	MOVQ	#1, FLAG	1657
			50 DD 00045 2\$:	PUSHL	FLAG	1668
			51 DD 00047	PUSHL	DEF_SYS	
			52 DD 00049	PUSHL	LOCK_REC	
		00000000V 00	03 FB 0004B	CALLS	#3, GET_USER_RECORD	

		03		50	E8	00052	BLBS	R0, 3\$		
				0169	31	00055	BRW	25\$		
		68		59	DD	00058	3\$: PUSHL	R9		1671
		14		01	FB	0005A	CALLS	#1, CLISPRESNT		
				50	E9	0005D	BLBC	R0, 4\$		
			5C	A7	9F	00060	PUSHAB	TOKENDSC		1672
				59	DD	00063	PUSHL	R9		
	00000000G	00		02	FB	00065	CALLS	#2, CLISGET_VALUE		
		05		50	E9	0006C	BLBC	R0, 4\$		
			5C	A7	B5	0006F	TSTW	TOKENLEN		1673
				08	12	00072	BNEQ	5\$		
				8F	DD	00074	4\$: PUSHL	#UAF\$_NOUSERNAME		1674
				37	11	0007A	BRB	9\$		
		0C		A7	B1	0007C	5\$: CMPW	TOKENLEN, #12		1680
			5C	09	1B	00080	BLEQU	6\$		
				8F	DD	00082	PUSHL	#UAF\$_NAMETOOBIG		1681
		68		01	FB	00088	CALLS	#1, LIBSSIGNAL		
		53		A7	3C	0008B	6\$: MOVZWL	TOKENLEN, R3		1686
			5C	53	D7	0008F	DECL	R3		
				52	D4	00091	CLRL	I		
				22	11	00093	BRB	11\$		
		51		C9	9A	00095	7\$: MOVZBL	SYMBOL_STR, R1		1688
		50		A7	D0	0009A	MOVL	TOKENPTR, R0		1690
		51		6240	3A	0009E	LOCC	(I)[R0], R1, SYMBOL_STR+1		
				02	12	000A5	BNEQ	8\$		
				51	D4	000A7	CLRL	R1		
				51	D5	000A9	8\$: TSTL	R1		
				08	12	000AB	BNEQ	10\$		
				8F	DD	000AD	PUSHL	#UAF\$_INVUSERNAME		1691
				78	11	000B3	9\$: BRB	14\$		
				52	D6	000B5	10\$: INCL	I		1690
		53		52	D1	000B7	11\$: CMPL	I, R3		
				D9	1B	000BA	BLEQU	7\$		
		10		67	E9	000BC	BLBC	UAF\$GL_CTLMSK, 12\$		1696
		56		A7	3C	000BF	MOVZWL	TOKENLEN, R6		1699
		50		A7	D0	000C3	MOVL	TOKENPTR, R0		
	6A	60		56	28	000C7	MOVC3	R6, (R0), NEWUSERNAME		
		AA		56	D0	000CB	MOVL	R6, NEWUSERLEN		1700
	6E	0080		20	28	000CF	12\$: MOVC3	#32, RECBUF+4, OLD_USER_BUFFER		1706
		50		A7	D0	000D5	MOVL	TOKENPTR, R0		1707
20	20	60		A7	2C	000D9	MOVC5	TOKENLEN, (R0), #32, #32, RECBUF+4		1708
				C7		000DF				
		06F4		01	D0	000E2	MOVL	#1, PWD_FLAG		1709
		00000000G		00	FB	000E7	CALLS	#0, UPDATE_RECORD		1711
				4D	50	000EE	BLBC	STATUS, 16\$		1712
				14	67	000F1	BLBS	UAF\$GL_CTLMSK, 13\$		1718
				C7	B4	000F4	CLRW	RECBUF+356		1721
08	00	6E		00	2C	000F8	MOVC5	#0, (SP), #0, #8, RECBUF+396		1722
				C7		000FD				
08	00	6E		00	2C	00100	MOVC5	#0, (SP), #0, #8, RECBUF+404		1723
				C7		00105				
				0210	C7	00108	13\$: PUSHAB	UAFRAB		1729
				0664	01	FB	CALLS	#1, SYSSPUT		
		00000000G		50	D0	00113	MOVL	R0, RMSERR		
		74		50	E8	00117	BLBS	R0, 17\$		
				A7	D0	0011A	MOVL	RMSERR, R0		1732
		000184EC		50	D1	0011E	CMPL	R0, #99564		

		0A	12	00125	BNEQ	15\$		
		8F	DD	00127	PUSHL	#UAF\$ UAEERR		1734
	68	01	FB	0012D	CALLS	#1, LIB\$SIGNAL		
		04	00130	RET				1736
		50	DD	00131	PUSHL	R0		1737
		7E	D4	00133	CLRL	-(SP)		
		8F	DD	00135	PUSHL	#UAF\$ ADDERR		
	68	03	FB	0013B	CALLS	#3, LIB\$SIGNAL		
		0080	31	0013E	BRW	25\$		1738
CC	AA	01	D0	00141	MOVL	#1, MODIFY_FLAG		1746
		5E	DD	00145	PUSHL	SP		1747
	08	67	E8	00147	BLBS	UAF\$GL CTLMSK, 18\$		
		8F	DD	0014A	PUSHL	#262148		
		06	11	00150	BRB	19\$		
		8F	DD	00152	PUSHL	#327682		
00000000V	00	02	FB	00158	CALLS	#2, SECURITY_AUDIT		
	58	67	E8	0015F	BLBS	UAF\$GL CTLMSR, 24\$		1751
		8F	DD	00162	PUSHL	#UAF\$ TOPMSG		1754
	68	01	FB	00168	CALLS	#1, LIB\$SIGNAL		
11	67	03	E1	0016B	BBC	#3, UAF\$GL CTLMSK, 20\$		1755
00	67	04	E0	0016F	BBS	#4, UAF\$GL CTLMSK, 20\$		1756
09	67	05	E1	00173	BBC	#5, UAF\$GL CTLMSK, 20\$		1757
		8F	DD	00177	PUSHL	#UAF\$ CLIWARNMSG		1758
	68	01	FB	0017D	CALLS	#1, LIB\$SIGNAL		
	09	C7	E9	00180	BLBC	PWD FLAG, 21\$		1763
		8F	DD	00185	PUSHL	#UAF\$ DEFPWD		1764
	68	01	FB	0018B	CALLS	#1, LIB\$SIGNAL		
	67	04	8A	0018E	BICB2	#4, UAF\$GL CTLMSK		1765
		34	A9	9F	PUSHAB	SD_ADD_IDENTIFIER		1766
	68	01	FB	00194	CALLS	#1, CLISPRESNT		
	23	50	E9	00197	BLBC	R0, 24\$		
	1F	70	A7	E9	BLBC	RDB EXISTS, 24\$		1767
		80	AA	9F	PUSHAB	SD_ATTRIBUTES		1774
	68	01	FB	001A1	CALLS	#1, CLISPRESNT		
	09	50	E9	001A4	BLBC	R0, 22\$		
00000000G	00	01	D0	001A7	MOVL	#1, ATTRIBUTES		1775
		06	11	001AE	BRB	23\$		
		00	D4	001B0	CLRL	ATTRIBUTES		1776
00000000G	00	00	FB	001B6	CALLS	#0, UAF\$ADD_IDENT_RECBUF		1777
	50	01	D0	001BD	MOVL	#1, R0		1791
		04	001C0	RET				
		50	D4	001C1	CLRL	R0		1793
		04	001C3	RET				

; Routine Size: 452 bytes. Routine Base: \$CODE\$ + 068D

create\_proxy - create NETUAF.DAT proxy file

```
1707 1794 1 %sbtll 'create_proxy - create NETUAF.DAT proxy file'
1708 1795 1 global routine create_proxy : novalue =
1709 1796 2 begin
1710 1797 1
1711 1798 1 ++
1712 1799 1
1713 1800 1
1714 1801 1 FUNCTIONAL DESCRIPTION:
1715 1802 1
1716 1803 1 This routine will create a DECnet Proxy Login File,
1717 1804 1 if and only if it does not already exist,
1718 1805 1 called NETUAF.DAT, in order to map remote users into
1719 1806 1 local accounts.
1720 1807 1
1721 1808 1 INPUTS:
1722 1809 1
1723 1810 1 none
1724 1811 1
1725 1812 1 OUTPUTS:
1726 1813 1
1727 1814 1 none
1728 1815 1
1729 1816 1 IMPLICIT INPUTS:
1730 1817 1
1731 1818 1 none
1732 1819 1
1733 1820 1 IMPLICIT OUTPUTS:
1734 1821 1
1735 1822 1 none
1736 1823 1
1737 1824 1 SIDE EFFECTS:
1738 1825 1
1739 1826 1 NETUAF.DAT is created and initialized
1740 1827 1
1741 1828 1 --
1742 1829 1
1743 1830 1
1744 1831 1 NETUAF.DAT should not already exist
1745 1832 1
1746 1833 1 if .netuaf_exists
1747 1834 1 then
1748 1835 1 begin
1749 1836 1 LIB$SIGNAL(UAF$_NAFAEX);
1750 1837 1 return;
1751 1838 1 end;
1752 1839 1
1753 1840 1
1754 1841 1 Should be able to create NETUAF.DAT with no problems
1755 1842 1
1756 1843 1 if rmsbad ($create (fab = naffab))
1757 1844 1 then LIB$SIGNAL(UAF$_NAFCREERR, 0, .rmserr);
1758 1845 1
1759 1846 1
1760 1847 1 Should connect ok, too
1761 1848 1
1762 1849 1 if rmsbad ($connect (rab = nafrab))
1763 1850 2 then LIB$SIGNAL(UAF$_NAFCONERR, 0, .rmserr);
```



create\_proxy - create NETUAF.DAT proxy file

```
1764 1851 2
1765 1852 2
1766 1853 2
1767 1854 2
1768 1855 2
1769 1856 2
1770 1857 2
1771 1858 2
1772 1859 2
1773 1860 2
1774 1861 2
1775 1862 2
1776 1863 2
1777 1864 2
1778 1865 2
1779 1866 2
1780 1867 2
1781 1868 2
1782 1869 2
```

```

Normal access is by key
nafrab [rab$b_rac] = rab$c_key;

Establish proper addresses
nafrab [rab$l_ubf] = netbuf;
nafrab [rab$l_rbf] = netbuf;

Set NETUAF.DAT existence flag
netuaf_exists = true;
netuaf_modified = true;

end;
```

			000C	00000		.ENTRY	CREATE PROXY, Save R2,R3		1795
	53	00000000G	00	9E	00002	MOVAB	LIB\$SIGNAL, R3		
	52	00000000'	00	9E	00009	MOVAB	RMSERR, R2		
	0A	F8	A2	E9	00010	BLBC	NETUAF EXISTS, 1\$		1833
		00000000G	8F	DD	00014	PUSHL	#UAF\$ NAFAEX		1836
	63		01	FB	0001A	CALLS	#1, LIB\$SIGNAL		
				04	0001D	RET			1835
		00000000'	00	9F	0001E	PUSHAB	NAFFAB		1843
	00000000G		01	FB	00024	CALLS	#1, SYSS\$CREATE		
	62		50	D0	0002B	MOVL	R0, RMSERR		
	0D		50	E8	0002E	BLBS	R0, 2\$		
			62	DD	00031	PUSHL	RMSERR		1844
			7E	D4	00033	CLRL	-(SP)		
		00000000G	8F	DD	00035	PUSHL	#UAF\$ NAFCREERR		
	63		03	FB	0003B	CALLS	#3, LIB\$SIGNAL		
		0634	C2	9F	0003E	PUSHAB	NAFRAB		1849
	00000000G		01	FB	00042	CALLS	#1, SYSS\$CONNECT		
	62		50	D0	00049	MOVL	R0, RMSERR		
	0D		50	E8	0004C	BLBS	R0, 3\$		
			62	DD	0004F	PUSHL	RMSERR		1850
			7E	D4	00051	CLRL	-(SP)		
		00000000G	8F	DD	00053	PUSHL	#UAF\$ NAFCONERR		
	63		03	FB	00059	CALLS	#3, LIB\$SIGNAL		
	0652	C2	01	90	0005C	MOVB	#1, NAFRAB+30		1855
	0658	C2	058C	9E	00061	MOVAB	NETBUF, NAFRAB+36		1860
	065C	C2	058C	9E	00068	MOVAB	NETBUF, NAFRAB+40		1861
	F8	A2	01	D0	0006F	MOVL	#1, NETUAF_EXISTS		1866
	00000000'	00	01	D0	00073	MOVL	#1, NETUAF_MODIFIED		1867
				04	0007A	RET			1869

; Routine Size: 123 bytes, Routine Base: \$CODE\$ + 0881

modify\_uaf - update user record (s)

```
1784 1870 1 %sbttl 'modify_uaf - update user record (s)'  
1785 1871 1 global routine modify_uaf : novalue =  
1786 1872 1 begin  
1787 1873 1  
1788 1874 1 ++  
1789 1875 1  
1790 1876 1 FUNCTIONAL DESCRIPTION:  
1791 1877 1  
1792 1878 1 Routine to modify any of the fields in one or more user records.  
1793 1879 1  
1794 1880 1 INPUTS:  
1795 1881 1  
1796 1882 1 none  
1797 1883 1  
1798 1884 1 IMPLICIT INPUTS:  
1799 1885 1  
1800 1886 1 none  
1801 1887 1  
1802 1888 1 OUTPUTS:  
1803 1889 1  
1804 1890 1 none  
1805 1891 1  
1806 1892 1 IMPLICIT OUTPUTS:  
1807 1893 1  
1808 1894 1 none  
1809 1895 1  
1810 1896 1 SIDE EFFECTS:  
1811 1897 1  
1812 1898 1 none  
1813 1899 1  
1814 1900 1 ROUTINE VALUE:  
1815 1901 1  
1816 1902 1 none  
1817 1903 1 --  
1818 1904 1  
1819 1905 1 local  
1820 1906 1 status;  
1821 1907 1  
1822 1908 1  
1823 1909 1 Obtain the user specification. This sets wildcard flags and initializes  
1824 1910 1 the appropriate key in RECBUF.  
1825 1911 1  
1826 1912 1  
1827 1913 1 if not parse_wild (sd_token1,false) ! Null string is disallowed  
1828 1914 1 then return;  
1829 1915 1  
1830 1916 1 ua:rab[ra$b1_rop] = ra$b1_rlk; ! Lock records for writing  
1831 1917 1 ra$bptr = out:rab;  
1832 1918 1 found_match = false;  
1833 1919 1  
1834 1920 1 if rmsbad (status = wild_user (modify_rec)) ! Modify each record  
1835 1921 1 then  
1836 1922 1 begin  
1837 1923 1 if .rmserr eql rms$rnf  
1838 1924 1 then  
1839 1925 1 LIB$SIGNAL(UAF$_BADSPC)  
1840 1926 1 else
```

modify\_uaf - update user record (s)

```
1841      (if .rmserr neq 0
1842      then
1843          LIB$SIGNAL(UAF$_MDFYERR, 0, .rmserr))
1844      end
1845      else
1846      begin
1847          if .status and .found_match
1848          then
1849              begin
1850                  LIB$SIGNAL(UAF$_MDFYMSG);
1851                  if (.uaf$gl_ctlmsk[uaf$y_cli]
1852                  and not .uaf$gl_ctlmsk[uaf$y_clitables])
1853                  and .uaf$gl_ctlmsk[uaf$y_clitab_pres]
1854                  then LIB$SIGNAL(UAF$_CLIWARNMSG);
1855                  end;
1856          end;
1857      end;
```

		001C 00000	.ENTRY	MODIFY UAF, Save R2,R3,R4	1871
	54 00000000G	00 9E 00002	MOVAB	LIB\$SIGNAL, R4	
	53 00000000'	00 9E 00009	MOVAB	UAF\$GL_CTLMSK, R3	
		7E D4 00010	CLRL	-(SP)	1913
	00000000'	00 9F 00012	PUSHAB	SD_TOKEN1	
		02 FB 00018	CALLS	#2, PARSE_WILD	
	0668 71	50 E9 0001F	BLBC	R0, 4\$	
	F4 A3 00080000	8F D0 00022	MOVL	#524288, UAFRAB+4	1916
		C3 9E 0002B	MOVAB	OUTRAB, RABPTR	1917
		C3 D4 00031	CLRL	FOUND_MATCH	1918
	00000000V	00 9F 00035	PUSHAB	MODIFY_REC	1920
	00000000V	01 FB 0003B	CALLS	#1, WILD_USER	
	74 A3	50 D0 00042	MOVL	STATUS, RMSERR	
	27	50 E8 00046	BLBS	STATUS, 2\$	
	52 74	A3 D0 00049	MOVL	RMSERR, R2	1923
	000182B2 8F	52 D1 0004D	CMPL	R2, #98994	
		08 12 00054	BNEQ	1\$	
	00000000G	8F DD 00056	PUSHL	#UAF\$_BADSPC	1925
		32 11 0005C	BRB	3\$	
		52 D5 0005E 1\$:	TSTL	R2	1927
		31 13 00060	BEQL	4\$	
		52 DD 00062	PUSHL	R2	1929
		7E D4 00064	CLRL	-(SP)	
	00000000G	8F DD 00066	PUSHL	#UAF\$_MDFYERR	
	64	03 FB 0006C	CALLS	#3, LIB\$SIGNAL	
		04 0006F	RET		1922
	1E 073C	C3 E9 00070 2\$:	BLBC	FOUND_MATCH, 4\$	1933
	00000000G	8F DD 00075	PUSHL	#UAF\$_MDFYMSG	1936
	64	01 FB 0007B	CALLS	#1, LIB\$SIGNAL	
11	63	03 E1 0007E	BBC	#3, UAF\$GL_CTLMSK, 4\$	1937
0D	63	04 E0 00082	BBS	#4, UAF\$GL_CTLMSK, 4\$	1938
09	63	05 E1 00086	BBC	#5, UAF\$GL_CTLMSK, 4\$	1939
	00000000G	8F DD 0008A	PUSHL	#UAF\$_CLIWARNMSG	1940
	64	01 FB 00090 3\$:	CALLS	#1, LIB\$SIGNAL	
		04 00093 4\$:	RET		1943

UAFMAIN  
V04-000

modify\_uaf - update user record (s)

K 15  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 69  
(10)

; Routine Size: 148 bytes, Routine Base: \$CODE\$ + 08FC



```
1859 1944 1 %sbttl 'modify_rec - update a user record action routine'
1860 1945 1 routine modify_rec =
1861 1946 2 begin
1862 1947 3
1863 1948 4 ++
1864 1949 5
1865 1950 6 FUNCTIONAL DESCRIPTION:
1866 1951 7
1867 1952 8     Modify an individual user record.
1868 1953 9
1869 1954 10 INPUTS:
1870 1955 11
1871 1956 12     none
1872 1957 13
1873 1958 14 IMPLICIT INPUTS:
1874 1959 15
1875 1960 16     RABPTR - RMS data structure for the file
1876 1961 17
1877 1962 18 OUTPUTS:
1878 1963 19
1879 1964 20     none
1880 1965 21
1881 1966 22 IMPLICIT OUTPUTS:
1882 1967 23
1883 1968 24     none
1884 1969 25
1885 1970 26 SIDE EFFECTS:
1886 1971 27
1887 1972 28     none
1888 1973 29
1889 1974 30 ROUTINE VALUE:
1890 1975 31
1891 1976 32     If an error is encountered the appropriate status is returned
1892 1977 33     except if the error message has already been output in which
1893 1978 34     case 0 is returned.
1894 1979 35 --
1895 1980 36
1896 1981 37 local
1897 1982 38     old_uic      : $bblock[4],
1898 1983 39     new_uic      : $bblock[4],
1899 1984 40     oldaccname   : vector [uaf$s_account,byte],
1900 1985 41     newaccname   : vector [uaf$s_account,byte],
1901 1986 42     oldaccdesc   : statdesc ,
1902 1987 43     newaccdesc   : statdesc ,
1903 1988 44     status       : long ;
1904 1989 45
1905 1990 46
1906 1991 47 User record has been read into RECBUF by caller. Update values
1907 1992 48 and modify the record.
1908 1993 49
1909 1994 50 When accessing records by uic, this routine is called repeatedly
1910 1995 51 from WILD_USER. UPDATE_RECORD is called to modify the appropriate
1911 1996 52 record fields for each requested record, and therefore must
1912 1997 53 reparse the command line each time. If call_count is greater
1913 1998 54 than 0, the command line is reparsed.
1914 1999 55
1915 2000 56
```

modify\_rec - update a user record action routin

```
1916 2001 2 IF .by_account
1917 2002 2 THEN
1918 2003 2 (IF NOT fmg$match_name (NAMELEN (UAF$$ ACCOUNT,UAF$T ACCOUNT),
1919 2004 2 RECBUF[UAF$T ACCOUNT],
1920 2005 2 .match_tokenlen, match_token)
1921 2006 2 THEN
1922 2007 2 RETURN TRUE)
1923 2008 2 ELSE
1924 2009 2 if .str_wild
1925 2010 2 and not fmg$match_name (namelen (uaf$$ username,recbuf[uaf$t_username]),
1926 2011 2 recbuf[uaf$t_username],
1927 2012 2 .match_tokenlen, match_token)
1928 2013 2 then return true;
1929 2014 2 found_match = true;
1930 2015 2
1931 2016 2 if ch$eq1 (.defuser<0,8>, defuser+1, .tokenlen, .tokenptr, ' ')
1932 2017 2 or ch$eq1 (.defuser<0,8>, defuser+1,
1933 2018 2 namelen (uaf$$ username, recbuf[uaf$t_username]),
1934 2019 2 recbuf[uaf$t_username], ' ')
1935 2020 2 then
1936 2021 2 begin
1937 2022 2 mod_default = true;
1938 2023 2 default_uaf ();
1939 2024 2 call_count = .call_count + 1;
1940 2025 2 return true;
1941 2026 2 end;
1942 2027 2
1943 2028 2
1944 2029 2 Save the old UIC and account name
1945 2030 2
1946 2031 2 old_uic[uic$v_format] = 0 ;
1947 2032 2 old_uic[uic$v_group] = .recbuf [uaf$w_grp] ;
1948 2033 2 old_uic[uic$v_member] = .recbuf [uaf$w_mem] ;
1949 2034 2 ch$move ( uaf$$ account, recbuf[uaf$t_account], oldaccname ) ;
1950 2035 2 oldaccdesc [length] = namelen ( uaf$$ account, recbuf[uaf$t_account]) ;
1951 2036 2 oldaccdesc [pointer] = oldaccname ;
1952 2037 2
1953 2038 2 if update_record ()
1954 2039 2 then
1955 2040 2 begin
1956 2041 2
1957 2042 2 Save the new UIC and account name
1958 2043 2
1959 2044 2 new_uic[uic$v_format] = 0 ;
1960 2045 2 new_uic[uic$v_group] = .recbuf [uaf$w_grp] ;
1961 2046 2 new_uic[uic$v_member] = .recbuf [uaf$w_mem] ;
1962 2047 2 ch$move ( uaf$$ account, recbuf[uaf$t_account], newaccname ) ;
1963 2048 2 newaccdesc [length] = namelen ( uaf$$ account, recbuf[uaf$t_account]) ;
1964 2049 2 newaccdesc [pointer] = newaccname ;
1965 2050 2
1966 2051 2
1967 2052 2 Update the UAF record
1968 2053 2
1969 2054 2 if rmsbad ($update (rab = uaf$rab))
1970 2055 2 then
1971 2056 2 begin
1972 2057 2 LIB$SIGNAL(UAF$ MDFYERR, 0, .rmserr);
```

modify\_rec - update a user record action routin

```
1973 2058 4      return .rmserr
1974 2059 4      end
1975 2060 3      else
1976 2061 4      begin
1977 2062 4      modify_flag = true;
1978 2063 4      security_audit (nsa$k_recid sysuaf_mod);
1979 2064 4      call_count = .call_count + 1;
1980 2065 3      end;
1981 2066 3
1982 2067 4      if (cli$present ( sd_modify_identifier ) and
1983 2068 4      .rdb_exists )
1984 2069 3      then
1985 2070 4      begin
1986 2071 4
1987 2072 4      |
1988 2073 4      | If the UIC changed then modify the identifiers
1989 2074 4      |
1990 2075 4      if .old_uic neq .new_uic
1991 2076 4      then
1992 2077 5      begin
1993 2078 5      local
1994 2079 5      oldidname      : vector [kgb$s_name, byte],
1995 2080 5      oldiddesc      : statdesc ;
1996 2081 5
1997 2082 5      oldiddesc[length] = kgb$s_name ;
1998 2083 5      oldiddesc[pointer] = oldidname ;
1999 2084 5      status = $idtoasc ( id      = .old_uic,
2000 2085 5      namlen = oldiddesc[length],
2001 2086 5      nambuf = oldiddesc ) ;
2002 2087 5
2003 2088 5      if not .status
2004 2089 5      then LIB$SIGNAL(UAF$_RDBMDFYERRU, 2,
2005 2090 5      .old_uic[uic$v_group],
2006 2091 5      .old_uic[uic$v_member], .status)
2007 2092 5
2008 2093 5      else if .status
2009 2094 5      then
2010 2095 5      begin
2011 2096 5      status = $mod_ident ( id      = .old_uic,
2012 2097 5      new_value = .new_uic ) ;
2013 2098 5
2014 2099 5      if not .status
2015 2100 5      then LIB$SIGNAL(UAF$_RDBMDFYERR, 1, oldiddesc, .status)
2016 2101 5      else
2017 2102 5      begin
2018 2103 5      rightslist_modified = true ;
2019 2104 5      LIB$SIGNAL(UAF$_RDBMDFYMSG, 1, oldiddesc);
2020 2105 5      end ;
2021 2106 5      end ;
2022 2107 5      end;
2023 2108 4      return true ;
2024 2109 4      end
2025 2110 3      else
2026 2111 4      begin
2027 2112 4      $release (rab = uaf$rab);
2028 2113 4      return false
2029 2114 4      end
```

.EXTRN SYSSUPDATE, SYSSMOD\_IDENT  
.EXTRN SYSSRELEASE

07FC 00000 MODIFY\_REC:

			5A	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	1945	
			59	000000000	00	9E	00009	MOVAB	LIBSSIGNAL, R10		
			58	000000000	00	9E	00010	MOVAB	DEFUSER+1, R9		
			5E	88	AE	9E	00017	MOVAB	RECBUF+4, R8		
			30	AE	010E0000	8F	D0	00018	MOVL	-120(SP), SP	
				34	AE	D4	00023	CLRL	#17694720, OLDACCDISC	1986	
			28	AE	010E0000	8F	D0	00026	MOVL	OLDACCDISC+4	
				2C	AE	D4	0002E	CLRL	#17694720, NEWACCDISC	1987	
				1F	06CC	C8	E9	00031	BLBC	NEWACCDISC+4	
			55	8C	A8	9E	00036	BLBC	STR WILD, 1\$	2009	
			53		68	9E	0003A	MOVAB	MATCH TOKEN, R5	2011	
	68		20		20	3A	0003D	MOVAB	RECBUF+4, R3		
			52	E0	A0	9E	00041	LOCC	#32, #32, RECBUF+4	2010	
			52		52	CE	00045	MOVAB	-32(R0), R2		
			54	D0	A8	D0	00048	MNEGL	R2, R2		
				00000000G	00	16	0004C	MOVL	MATCH TOKENLEN, R4	2011	
			37		50	E9	00052	JSB	FMG\$MATCH_NAME		
	06BC		C8		01	D0	00055	BLBC	R0, 3\$		
			54	FF	A9	9A	0005A	MOVL	#1, FOUND MATCH	2014	
			50	E0	A8	D0	0005E	MOVZBL	DEFUSER, R4	2016	
DC	A8	20	69		54	2D	00062	MOVL	TOKENPTR, R0		
					60		00068	CMPC5	R4, DEFUSER+1, #32, TOKENLEN, (R0)		
					10	13	00069	BEQL	2\$		
		68	20		20	3A	0006B	LOCC	#32, #32, RECBUF+4	2018	
		50	20		50	C3	0006F	SUBL3	R0, #32, R0		
50		20	69		54	2D	00073	CMPC5	R4, DEFUSER+1, #32, R0, RECBUF+4	2019	
					68		00078				
					14	12	00079	BNEQ	4\$		
		000000000	00		01	D0	0007B	MOVL	#1, MOD DEFAULT	2022	
		00000000V	00		00	FB	00082	CALLS	#0, DEFAULT_UAF	2023	
				D8	A8	D6	00089	INCL	CALL_COUNT	2024	
					0122	31	0008C	BRW	10\$	2025	
55	02	1E			00	F0	0008F	INSV	#0, #30, #2, OLD UIC	2031	
56	0E	10			22	A8	00094	INSV	RECBUF+38, #16, #14, OLD_UIC	2032	
		56			20	A8	0009A	MOVW	RECBUF+36, OLD_UIC	2033	
					20	28	0009E	MOVW	RECBUF+36, OLD_UIC	2033	
	58	AE	30		20	28	0009E	MOVW	RECBUF+36, OLD_UIC	2033	
	30	A8			20	3A	000A4	LOCC	#32, RECBUF+52, OLDACCDISC	2034	
	30	AE			20	3A	000A4	LOCC	#32, #32, RECBUF+52	2035	
					50	A3	000A9	SUBW3	R0, #32, OLDACCDISC		
		34			58	AE	000AE	MOVAB	OLDACCDISC, OLDACCDISC+4	2036	
		00000000G	00		00	FB	000B3	CALLS	#0, UPDATE_RECORD	2038	
			03		50	E8	000BA	BLBS	R0, 5\$		
					00F5	31	000BD	BRW	11\$		
57	02	1E			00	F0	000C0	INSV	#0, #30, #2, NEW UIC	2044	
57	0E	10			22	A8	000C5	INSV	RECBUF+38, #16, #14, NEW_UIC	2045	
		57			20	A8	000CB	MOVW	RECBUF+36, NEW_UIC	2046	
					20	28	000CF	MOVW	RECBUF+36, NEW_UIC	2046	
	38	AE	30		20	3A	000D5	LOCC	#32, RECBUF+52, NEWACCDISC	2047	
	30	A8			20	3A	000D5	LOCC	#32, #32, RECBUF+52	2048	
	28	AE			50	A3	000DA	SUBW3	R0, #32, NEWACCDISC		



2C	AE	38	AE	9E	000DF	MOVAB	NEWACNAME, NEWACDESC+4	2049
		05E4	C8	9F	000E4	PUSHAB	UAFRAB	2054
00000000G	00		01	FB	000E8	CALLS	#1, SYSSUPDATE	
F4	A8		50	D0	000EF	MOVL	R0, RMSERR	
	13		50	E8	000F3	BLBS	R0, 6\$	
		F4	A8	DD	000F6	PUSHL	RMSERR	2057
			7E	D4	000F9	CLRL	-(SP)	
		00000000G	8F	DD	000FB	PUSHL	#UAF\$ MDFYERR	
	6A		03	FB	00101	CALLS	#3, LIBSSIGNAL	
	50	F4	A8	D0	00104	MOVL	RMSERR, R0	2058
				04	00108	RET		
00000000'	00		01	D0	00109	MOVL	#1, MODIFY_FLAG	2062
		00030002	8F	DD	00110	PUSHL	#16610	2063
00000000V	00		01	FB	00116	CALLS	#1, SECURITY_AUDIT	
		D8	A8	D6	0011D	INCL	CALL COUNT	2064
		FF63	C9	9F	00120	PUSHAB	SD MODIFY IDENTIFIER	2067
00000000G	00		01	FB	00124	CALLS	#1, CLISPRESENT	
	70		50	E9	0012B	BLBC	R0, 8\$	
	7F	F0	A8	E9	0012E	BLBC	RDB EXISTS, 10\$	2068
	57		56	D1	00132	CMP	OLD_UIC, NEW_UIC	2075
			7A	13	00135	BEQL	10\$	
	6E	010E0000	8F	D0	00137	MOVL	#17694720, OLDDIDDESC	2080
		04	AE	D4	0013E	CLRL	OLDDIDDESC+4	
	6E		20	B0	00141	MOVW	#32, OLDDIDDESC	2082
04	AE	08	AE	9E	00144	MOVAB	OLDIDNAME, OLDDIDDESC+4	2083
			7E	7C	00149	CLRL	-(SP)	2086
			7E	D4	0014B	CLRL	-(SP)	
		0C	AE	9F	0014D	PUSHAB	OLDDIDDESC	
		10	AE	9F	00150	PUSHAB	OLDDIDDESC	
			56	DD	00153	PUSHL	OLD_UIC	
00000000G	00		06	FB	00155	CALLS	#6, SYSSIDTOASC	
	52		50	D0	0015C	MOVL	R0, STATUS	
	17		52	E8	0015F	BLBS	STATUS, 7\$	2088
			52	DD	00162	PUSHL	STATUS	2091
	7E		56	3C	00164	MOVZWL	OLD_UIC, -(SP)	
7E	0E		10	EF	00167	EXTZV	#16, #14, OLD_UIC, -(SP)	2090
			02	DD	0016C	PUSHL	#2	2089
		00000000G	8F	DD	0016E	PUSHL	#UAF\$ RDBMDFYERR	
	6A		05	FB	00174	CALLS	#5, LIBSSIGNAL	
			38	11	00177	BRB	10\$	
			57	DD	00179	PUSHL	NEW UIC	2096
			7E	7C	0017B	CLRL	-(SP)	
			7E	D4	0017D	CLRL	-(SP)	
			56	DD	0017F	PUSHL	OLD_UIC	
00000000G	00		05	FB	00181	CALLS	#5, SYSSMOD_IDENT	
	52		50	D0	00188	MOVL	R0, STATUS	
	12		52	E8	0018B	BLBS	STATUS, 9\$	2098
			52	DD	0018E	PUSHL	STATUS	2099
		04	AE	9F	00190	PUSHAB	OLDDIDDESC	
			01	DD	00193	PUSHL	#1	
		00000000G	8F	DD	00195	PUSHL	#UAF\$ RDBMDFYERR	
	6A		04	FB	0019B	CALLS	#4, LIBSSIGNAL	
			11	11	0019E	BRB	10\$	
F8	A8		01	90	001A0	MOVW	#1, RIGHTSLIST_MODIFIED	2102
			5E	DD	001A4	PUSHL	SP	2103
			01	DD	001A6	PUSHL	#1	
		00000000G	8F	DD	001A8	PUSHL	#UAF\$ RDBMDFYMSG	

UAFMAIN  
V04-000

D 16  
16-Sep-1984 02:16:54 VAX-11 Bliss-32 V4.0-742 Page 75  
modify\_rec - update a user record action routin 14-Sep-1984 13:21:22 DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 (11)

6A	03	FB	001AE		CALLS	#3, LIB\$SIGNAL	:		
50	01	D0	001B1	10\$:	MOVL	#1, R0	:	2111	
		04	001B4		RET		:		
00000000G 00	05E4	C8	9F	001B5	11\$:	PUSHAB	UAFRAB	:	2112
		01	FB	001B9		CALLS	#1, SYS\$RELEASE	:	
		50	D4	001C0		CLRL	R0	:	2113
		04	001C2		RET		:	2115	

; Routine Size: 451 bytes. Routine Base: \$CODE\$ + 0990

remove\_uaf - remove username from file

```
2032 2116 1 %sbtcl 'remove_uaf - remove username from file'
2033 2117 1 global routine remove_uaf : novalue =
2034 2118 2 begin
2035 2119 2
2036 2120 2 ++
2037 2121 2
2038 2122 2 FUNCTIONAL DESCRIPTION:
2039 2123 2
2040 2124 2 Routine to delete a user record from the UAF file.
2041 2125 2
2042 2126 2 INPUTS:
2043 2127 2
2044 2128 2 none
2045 2129 2
2046 2130 2 OUTPUTS:
2047 2131 2
2048 2132 2 none
2049 2133 2
2050 2134 2 IMPLICIT INPUTS:
2051 2135 2
2052 2136 2 rename: a logical flag which indicates whether this COPY is part
2053 2137 2 of a RENAME operation
2054 2138 2
2055 2139 2 ROUTINE VALUE:
2056 2140 2
2057 2141 2 none
2058 2142 2
2059 2143 2 SIDE EFFECTS:
2060 2144 2
2061 2145 2 This routine also deletes any NETUAF entries for the
2062 2146 2 local user which is to be removed (If NETUAF.DAT exists)
2063 2147 2
2064 2148 2 --
2065 2149 2
2066 2150 2
2067 2151 2 Look for the record specified by the user. Make sure
2068 2152 2 the DEFAULT and SYSTEM records are not removed.
2069 2153 2
2070 2154 2
2071 2155 2 if get_user_record (true, false)
2072 2156 2 then
2073 2157 2
2074 2158 2
2075 2159 2 User record has been found and read into RECBUF
2076 2160 2 Zero the username field and update the record in the file
2077 2161 2
2078 2162 2 begin
2079 2163 2 if rmsbad ($delete (rab=uafrab))
2080 2164 2 then
2081 2165 2 LIB$SIGNAL(UAF$_REMERR, 0, .rmserr)
2082 2166 2 else
2083 2167 2 begin
2084 2168 2 modify_flag = true; ! mark file as modified
2085 2169 2 if not .uaf$gl_ctlmsk[uaf$v_rename]
2086 2170 2 then
2087 2171 2 security_audit (nsa$recid_sysuaf_del);
2088 2172 2 if not .uaf$gl_ctlmsk[uaf$v_rename]
```

remove\_uaf - remove username from file

```
2089      then
2090      begin
2091      If there is a proxy login file, delete entries for
2092      the user just removed from SYSUAF.DAT
2093      if .netuaf_exists
2094      then adjust_proxy (remove_records);
2095      LIB$SIGNAL(UAF$_REMSG);
2096      Unless specifically requested not to
2097      remove the corresponding identifier from the
2098      rights data base.
2099      if (cli$present ( sd_remove_identifier ) and
2100      .rdb_exists )
2101      then
2102      uaf$remove_ident_recbuf ( );
2103      end;
2104      end;
2105      end;
2106      end;
2107      end;
2108      end;
2109      end;
2110      end;
```

				000C 00000	.EXTRN  SYS\$DELETE	
				00 9E 00002	.ENTRY  REMOVE UAF, Save R2,R3	2117
				00 9E 00009	MOVAB  LIB\$SIGNAL, R3	
				01 7D 00010	MOVAB  RMSERR, R2	
00000000V	00			02 FB 00013	MOVQ   #1, -(SP)	2155
	6C			50 E9 0001A	CALLS  #2, GET_USER_RECORD	
		05F0		C2 9F 0001D	BLBC   R0, 3\$	
00000000G	00			01 FB 00021	PUSHAB UAF\$RAB	2163
	62			50 D0 00028	CALLS  #1, SYS\$DELETE	
	0E			50 E8 0002B	MOVL   R0, RMSERR	
				62 DD 0002E	BLBS   R0, 1\$	
				7E D4 00030	PUSHL  RMSERR	2165
				8F DD 00032	CLRL   -(SP)	
	63	00000000G		03 FB 00038	PUSHL  #UAF\$ REMERR	
				04 0003B	CALLS  #3, LIB\$SIGNAL	
00000000'	00			01 D0 0003C 1\$:	RET	
	42	8C		A2 E8 00043	MOVL   #1, MODIFY_FLAG	2168
		00020002		8F DD 00047	BLBS   UAF\$GL_CTLMSK, 3\$	2169
00000000V	00			01 FB 0004D	PUSHL  #131074	2171
	31	8C		A2 E8 00054	CALLS  #1, SECURITY_AUDIT	
	09	F8		A2 E9 00058	BLBS   UAF\$GL_CTLMSK, 3\$	2172
				01 DD 0005C	BLBC   NETUAF_EXISTS, 2\$	2179
00000000V	00			01 FB 0005E	PUSHL  #1	2180
		00000000G		8F DD 00065 2\$:	CALLS  #1, ADJUST_PROXY	
	63			01 FB 0006B	PUSHL  #UAF\$ REMMSG	2181
		00000000'		00 9F 0006E	CALLS  #1, LIB\$SIGNAL	
00000000G	00			01 FB 00074	PUSHAB SD_REMOVE_IDENTIFIER	2187
	08			50 E9 0007B	CALLS  #1, CLIPRESENT	
	07	FC		A2 E9 0007E	BLBC   R0, 3\$	
00000000G	00			00 FB 00082	BLBC   RDB_EXISTS, 3\$	2188
					CALLS  #0, -UAF\$REMOVE_IDENT_RECBUF	2190



UAFMAIN  
V04-000

remove\_uaf - remove username from file

G 16  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 (12)

Page 78

04 00089 3\$: RET

; 2194

; Routine Size: 138 bytes, Routine Base: \$CODE\$ + 0B53

remove\_proxy - remove a proxy record

```
2112 2195 1 %sbtcl 'remove_proxy - remove a proxy record'
2113 2196 1 global routine remove_proxy : novalue =
2114 2197 1 begin
2115 2198 1
2116 2199 1 ++
2117 2200 1
2118 2201 1 FUNCTIONAL CHARACTERISTICS:
2119 2202 1
2120 2203 1 This routine removes proxy login entries from NETUAF.DAT
2121 2204 1
2122 2205 1 INPUTS:
2123 2206 1
2124 2207 1 none
2125 2208 1
2126 2209 1 OUTPUTS:
2127 2210 1
2128 2211 1 none
2129 2212 1
2130 2213 1 IMPLICIT INPUTS:
2131 2214 1
2132 2215 1 none
2133 2216 1
2134 2217 1 IMPLICIT OUTPUTS:
2135 2218 1
2136 2219 1 none
2137 2220 1
2138 2221 1 ROUTINE VALUE:
2139 2222 1
2140 2223 1 none
2141 2224 1
2142 2225 1 SIDE EFFECTS:
2143 2226 1
2144 2227 1 An entry is removed from NETUAF.DAT
2145 2228 1
2146 2229 1 --
2147 2230 1
2148 2231 1 local
2149 2232 1 node_len,
2150 2233 1 node_ptr,
2151 2234 1 remuser_len,
2152 2235 1 remuser_ptr,
2153 2236 1 counter,
2154 2237 1 success;
2155 2238 1
2156 2239 1
2157 2240 1 Make sure NETUAF.DAT exists
2158 2241 1
2159 2242 1 if not .netuaf exists
2160 2243 1 then return LIB$SIGNAL(UAF$_NAFDNE);
2161 2244 1
2162 2245 1
2163 2246 1 Retrieve remote name in node::remotename form
2164 2247 1
2165 2248 1 cli$get_value (sd_token1, tokendsc);
2166 2249 1
2167 2250 1
2168 2251 1 Verify proper format
```

remove\_proxy - remove a proxy record

```
2169 2252 1
2170 2253 if not remote_parse (node_ptr, node_len, remuser_ptr, remuser_len)
2171 2254 then return;
2172 2255
2173 2256
2174 2257 Copy into appropriate fields
2175 2258
2176 2259 ch$copy (.node_len, .node_ptr, ' ', naf$s_node, netbuf[naf$t_node]);
2177 2260 ch$copy (.remuser_len, .remuser_ptr, ' ', naf$s_remuser, netbuf[naf$t_remuser]);
2178 2261
2179 2262 nafrab[rab$l_rop] = rab$m_rlk;
2180 2263
2181 2264 success = get_proxy_record ();
2182 2265
2183 2266
2184 2267 Delete the record
2185 2268
2186 2269 if .success
2187 2270 then
2188 2271 begin
2189 2272 if rmsbad ($delete (rab = nafrab))
2190 2273 then
2191 2274 LIB$SIGNAL(UAF$_REMERR, 0, .rmserr)
2192 2275 else
2193 2276 begin
2194 2277 security_audit (nsa$k_recid_netuaf_del);
2195 2278 LIB$SIGNAL(UAF$_PREMSG);
2196 2279 end;
2197 2280 end
2198 2281 else
2199 2282 LIB$SIGNAL(UAF$_REMERR);
2200 2283
2201 2284 netuaf_modified = true;
2202 2285 1
end;
```

```
58 00000000G 8F 01FC 00000
57 00000000G 00 9E 00002
56 00000000' 00 9E 00009
5E 10 C2 00017
0A F8 A6 E8 0001A
00000000G 8F DD 0001E
67 01 FB 00024
04 00027
E8 A6 9F 00028 1$:
00000000' 00 9F 0002B
00 02 FB 00031
SE DD 00038
08 AE 9F 0003A
10 AE 9F 0003D
18 AE 9F 00040
FA17 CF 04 FB 00043
63 50 E9 00048
```

```
.ENTRY REMOVE_PROXY, Save R2,R3,R4,R5,R6,R7,R8
2196
MOVL #UAF$_REMERR, R8
MOVAB LIB$SIGNAL, R7
MOVAB RMSERR, R6
SUBL2 #16, SP
BLBS NETUAF_EXISTS, 1$
2242
PUSHL #UAF$_RAFDNE
2243
CALLS #1, LIB$SIGNAL
RET
PUSHAB TOKENDSC
2248
PUSHAB SD_TOKEN1
CALLS #2, CLISGET_VALUE
PUSHL SP
2253
PUSHAB REMUSER_PTR
PUSHAB NODE_LEN
PUSHAB NODE_PTR
CALLS #4, REMOTE_PARSE
BLBC R0, 6$
```

UAFMAIN  
V04-000

remove\_proxy - remove a proxy record

J 16  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 81  
(13)

20	20	0C	BE	08	AE	2C	0004B	MOVCS	NODE_LEN, @NODE_PTR, #32, #32, NETBUF	: 2259
				058C	C6		00052			
20	20	04	BE		6E	2C	00055	MOVCS	REMUSER_LEN, @REMUSER_PTR, #32, #32, -	: 2260
				C5AC	C6		00058		NETBUF+32	
	0638	C6	00080000		8F	D0	0005E	MOVL	#524288, NAFRAB+4	: 2262
	00000000V	00			00	FB	00067	CALLS	#0, GET_PROXY_RECORD	: 2264
		31			50	E9	0006E	BLBC	SUCCESS, 3\$	: 2269
			0634		C6	9F	00071	PUSHAB	NAFRAB	: 2272
	00000000G	00			01	FB	00075	CALLS	#1, SYS\$DELETE	
		66			50	D0	0007C	MOVL	R0, RMSERR	
		0B			50	E8	0007F	BLBS	R0, 2\$	
					66	DD	00082	PUSHL	RMSERR	: 2274
					7E	D4	00084	CLRL	-(SP)	
					58	DD	00086	PUSHL	R8	
		67			03	FB	00088	CALLS	#3, LIB\$SIGNAL	
					1A	11	0008B	BRB	5\$	
			00020003		8F	DD	0008D	PUSHL	#131075	: 2277
	00000000V	00			01	FB	00093	CALLS	#1, SECURITY_AUDIT	
			00000000G		8F	DD	0009A	PUSHL	#UAF\$_PREMSG	: 2278
					02	11	000A0	BRB	4\$	
					58	DD	000A2	PUSHL	R8	: 2282
		67			01	FB	000A4	CALLS	#1, LIB\$SIGNAL	
	00000000'	00			01	D0	000A7	MOVL	#1, NETUAF_MODIFIED	: 2284
					04	000AE	6\$:	RET		: 2285

; Routine Size: 175 bytes, Routine Base: \$CODE\$ + 0BDD



rename\_uaf - rename user record

```
2204 2286 1 %sbttl 'rename_uaf - rename user record'
2205 2287 1 global routine rename_uaf : novalue =
2206 2288 2 begin
2207 2289 2
2208 2290 2 ++
2209 2291 2
2210 2292 2 FUNCTIONAL DESCRIPTION:
2211 2293 2
2212 2294 2     Effect a user authorization record rename, by
2213 2295 2     performing a COPY and a REMOVE operation.
2214 2296 2
2215 2297 2 INPUTS:
2216 2298 2
2217 2299 2     none
2218 2300 2
2219 2301 2 IMPLICIT INPUTS:
2220 2302 2
2221 2303 2     none
2222 2304 2
2223 2305 2 OUTPUTS:
2224 2306 2
2225 2307 2     none
2226 2308 2
2227 2309 2 IMPLICIT OUTPUTS:
2228 2310 2
2229 2311 2     none
2230 2312 2
2231 2313 2 SIDE EFFECTS:
2232 2314 2
2233 2315 2     A user authorization record is copied with a newname; the
2234 2316 2     original record is then deleted.
2235 2317 2     This routine also causes updating of any NETUAF entries for the
2236 2318 2     local user which is to be renamed.
2237 2319 2
2238 2320 2 --
2239 2321 2
2240 2322 2 uaf$gl_ctlmsk[uaf$v_rename] = true;
2241 2323 2
2242 2324 2 Copy the new authorization record
2243 2325 2
2244 2326 2 if (copy_uaf ())
2245 2327 2 then
2246 2328 2     begin
2247 2329 2
2248 2330 2 Remove the old authorization record
2249 2331 2
2250 2332 2     remove_uaf ();
2251 2333 2     LIB$SIGNAL(UAF$ RENMSG);
2252 2334 2
2253 2335 2 Because passwords are folded in with the username, passwords for
2254 2336 2 RENAMED records will no longer work--warn the user
2255 2337 2
2256 2338 2     if .pwd flag
2257 2339 2     then LIB$SIGNAL(UAF$ DEFPWD);
2258 2340 2
2259 2341 2
2260 2342 2 Modify the rights data base if one exists
```

rename\_uaf - rename user record

```
2261 2343 3 !
2262 2344 4 if (cli$present ( sd_modify_identifier ) and
2263 2345 4 .rdb_exists )
2264 2346 3 then
2265 2347 4 begin
2266 2348 4 local
2267 2349 4 status : long,
2268 2350 4 old_name_buff : vector [kgb$s_name, byte],
2269 2351 4 old_name_desc : statdesc,
2270 2352 4 new_name_desc : statdesc ;
2271 2353 4
2272 2354 4 old_name_desc[length] = kgb$s_name ;
2273 2355 4 old_name_desc[pointer] = old_name_buff ;
2274 2356 4 new_name_desc[length] = .newuser[en] ;
2275 2357 4 new_name_desc[pointer] = newusername ;
2276 2358 4
2277 2359 4 uaf$find_uic ( ) ; ! Build Identifier from the recbuf
2278 2360 4
2279 2361 4 !
2280 2362 4 ! Find the ascii name
2281 2363 4 !
2282 2364 4 status = $idtoasc ( id = .ident,
2283 2365 4 namlen = old_name_desc[length],
2284 2366 4 nambuf = old_name_desc ) ;
2285 2367 4
2286 2368 4 if not .status
2287 2369 4 then LIB$SIGNAL(UAF$_RDBMDFYERRU, 2,
2288 2370 4 .ident[uic$v_group],
2289 2371 4 .ident[uic$v_member], .status)
2290 2372 4 else
2291 2373 4 begin
2292 2374 4 status = $mod_ident ( id = .ident,
2293 2375 4 new_name = new_name_desc ) ;
2294 2376 4
2295 2377 4 if not .status
2296 2378 4 then LIB$SIGNAL(UAF$_RDBMDFYERR, 1, old_name_desc, .status)
2297 2379 4 else
2298 2380 4 begin
2299 2381 4 rightslist_modified = true ;
2300 2382 4 LIB$SIGNAL(UAF$_RDBMDFYMSG, 1, old_name_desc);
2301 2383 4 end ;
2302 2384 4 end ;
2303 2385 4 end ;
2304 2386 4
2305 2387 4 end;
2306 2388 4
2307 2389 4 !
2308 2390 4 ! Reset flags
2309 2391 4 rename_ph2 = false;
2310 2392 4 uaf$gl_ctlmsk[uaf$v_rename] = false;
2311 2393 4
2312 2394 4 end;
```

			007C	00000	.ENTRY	RENAME UAF, Save R2,R3,R4,R5,R6	2287	
	56	00000000'	00	9E	00002	MOVAB	NEWUSERLEN, R6	
	55	000000000G	00	9E	00009	MOVAB	IDENT, R5	
	54	000000000'	00	9E	00010	MOVAB	UAF\$GL CTLMSK, R4	
	53	000000000G	00	9E	00017	MOVAB	LIB\$SIGNAL, R3	
	5E		30	C2	0001E	SUBL2	#48, SP	
	64		01	88	00021	BISB2	#1, UAF\$GL CTLMSK	2322
FA08	CF		00	FB	00024	CALLS	#0, COPY_UAF	2326
	29		50	E9	00029	BLBC	R0, 2\$	
FE96	CF		00	FB	0002C	CALLS	#0, REMOVE UAF	2332
		00000000G	8F	DD	00031	PUSHL	#UAF\$ RENMSG	2333
	63		01	FB	00037	CALLS	#1, LIB\$SIGNAL	
	09	06F4	C4	E9	0003A	BLBC	PWD FLAG, 1\$	2338
		0000C000G	8F	DD	0003F	PUSHL	#UAF\$ DEFPWD	2339
	63		01	FB	00045	CALLS	#1, LIB\$SIGNAL	
		00000000'	00	9F	00048	PUSHAB	SD_MODIFY IDENTIFIER	2344
00000000G	00		01	FB	0004E	CALLS	#1, CLIPRESENT	
	60		50	E9	00055	BLBC	R0, 3\$	
	5C	70	A4	E9	00058	BLBC	RDB EXISTS, 3\$	2345
08	AE	010E0000	8F	DD	0005C	MOVL	#17894720, OLD_NAME_DESC	2351
		0C	AE	D4	00064	CLRL	OLD_NAME_DESC+4	
	6E	010E0000	8F	DD	00067	MOVL	#17894720, NEW_NAME_DESC	2352
		04	AE	D4	0006E	CLRL	NEW_NAME_DESC+4	
	08	AE	20	B0	00071	MOVW	#32, OLD_NAME_DESC	2354
	0C	AE	10	AE	00075	MOVAB	OLD_NAME_BUFF, OLD_NAME_DESC+4	2355
	6E		66	B0	0007A	MOVW	NEWUSERLEN, NEW_NAME_DESC	2356
04	AE	04	A6	9E	0007D	MOVAB	NEWUSERNAME, NEW_NAME_DESC+4	2357
00000000G	00		00	FB	00082	CALLS	#0, UAF\$FIND_UIC	2359
			7E	7C	00089	CLRQ	-(SP)	2366
			7E	D4	0008B	CLRL	-(SP)	
		14	AE	9F	0008D	PUSHAB	OLD_NAME_DESC	
		18	AE	9F	00090	PUSHAB	OLD_NAME_DESC	
			65	DD	00093	PUSHL	IDENT	
00000000G	00		06	FB	00095	CALLS	#6, SYS\$IDTOASC	
	52		50	DD	0009C	MOVL	R0, STATUS	
	18		52	E8	0009F	BLBS	STATUS, 4\$	2367
			52	DD	000A2	PUSHL	STATUS	2370
	7E		65	3C	000A4	MOVZWL	IDENT, -(SP)	
7E	0E		00	EF	000A7	EXTZV	#0, #14, IDENT+2, -(SP)	2369
			02	DD	000AD	PUSHL	#2	2368
		00000000G	8F	DD	000AF	PUSHL	#UAF\$ RDBMDFYERRU	
	63		05	FB	000B5	CALLS	#5, LIB\$SIGNAL	
			3A	11	000B8	BRB	6\$	
			7E	D4	000BA	CLRL	-(SP)	2375
		04	AE	9F	000BC	PUSHAB	NEW_NAME_DESC	
			7E	7C	000BF	CLRQ	-(SP)	
			65	DD	000C1	PUSHL	IDENT	
00000000G	00		05	FB	000C3	CALLS	#5, SYS\$MOD_IDENT	
	52		50	DD	000CA	MOVL	R0, STATUS	
	12		52	E8	000CD	BLBS	STATUS, 5\$	2377
			52	DD	000D0	PUSHL	STATUS	2378
		0C	AE	9F	000D2	PUSHAB	OLD_NAME_DESC	
			01	DD	000D5	PUSHL	#1	
		00000000G	8F	DD	000D7	PUSHL	#UAF\$ RDBMDFYERR	
	63		04	FB	000DD	CALLS	#4, LIB\$SIGNAL	
			12	11	000EO	BRB	6\$	
78	A4		01	90	000E2	MOVW	#1, RIGHTSLIST_MODIFIED	2381

UAFMAIN  
V04-000

rename\_uaf - rename user record

B 1  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 85  
(14)

	08	AE	9F	000E6	PUSHAB	OLD_NAME_DESC	:	2382
		01	DD	000E9	PUSHL	#1	:	
	00000000G	8F	DD	000EB	PUSHL	#UAF\$RDBMDFYMSG	:	
63		03	FB	000F1	CALLS	#3, LTB\$SIGNAL	:	
	D8	A6	94	000F4	CLRB	RENAME_PH2	:	2391
64		01	8A	000F7	BICB2	#1, UAF\$GL_CTLMSK	:	2392
			04	000FA	RET		:	2394

; Routine Size: 251 bytes,      Routine Base: \$CODE\$ + 0C8C

UA  
VO

```
2314 1 %sbttl 'adjust_proxy - implicitly remove/update proxy record'
2315 1 global routine adjust_proxy (remove) : novalue =
2316 begin
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
```

```
2395 1 %sbttl 'adjust_proxy - implicitly remove/update proxy record'
2396 1 global routine adjust_proxy (remove) : novalue =
2397 begin
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
```

```

++
FUNCTIONAL DESCRIPTION:
    This routine performs the operations implicitly indicated by
    REMOVE or RENAME operations on SYSUAF.DAT.  If a SYSUAF.DAT
    record is removed, then any corresponding NETUAF.DAT entries must
    also be deleted.  If a SYSUAF.DAT record is renamed, then any
    corresponding NETUAF.DAT entries must be updated.

INPUTS:
    remove - a flag which indicates that the NETUAF.DAT record
             should be removed.  If false, then the record
             should be updated.

OUTPUTS:
    none

IMPLICIT INPUTS:
    olduserlen - the old username length for a RENAME
    oldusername - the old username string for a RENAME
    TOKENPTR - the new username for RENAME, the removed username for REMOVE
    TOKENLEN - the new username length, the removed username length

IMPLICIT OUTPUTS:
    none

SIDE EFFECTS:
    A record is removed/updated in NETUAF.DAT

--
local
    modified: initial(false),
    status;

    Set access to sequential because we need to check for
    multiple entries

    nafrab[rab$l_rop] = rab$m_rlk;          ! Lock records for writing
    nafrab[rab$b_rac] = rab$c_seq;
    $rewind (rab = nafrab);

    Until EOF...

while status = get_proxy_record ()
```



```
2371 2452 2 do
2372 2453     begin
2373 2454     local
2374 2455         locuser_len,
2375 2456         blank_ptr;
2376 2457
2377 2458
2378 2459     Find end of user name...
2379 2460
2380 2461
2381 2462     If not found in 12 characters, it must be the full 12
2382 2463     characters in length
2383 2464
2384 2465     if ch$fail (blank_ptr = ch$find_ch (naf$s_localuser,
2385 2466                                         netbuf[naf$t_localuser], ' '))
2386 2467     then
2387 2468         locuser_len = naf$s_localuser
2388 2469     else
2389 2470         locuser_len = .blank_ptr - netbuf[naf$t_localuser];
2390 2471
2391 2472
2392 2473     If this is a record to be removed, delete it
2393 2474
2394 2475     if .remove
2395 2476     then
2396 2477         begin
2397 2478             if .tokenlen eql .locuser_len
2398 2479             and ch$eql (.tokenlen, .tokenptr, .locuser_len, netbuf[naf$t_localuser])
2399 2480             then
2400 2481                 begin
2401 2482                     netuaf_modified = true;
2402 2483                     $delete (rab = nafrab);
2403 2484                     security_audit (nsa$k_recid_netuaf_del);
2404 2485                     end;
2405 2486                 end
2406 2487
2407 2488     otherwise, change the localusername field to reflect the
2408 2489     new username in SYSUAF.DAT
2409 2490
2410 2491     else
2411 2492         begin
2412 2493             modified = true;
2413 2494             if .olduserlen eql .locuser_len
2414 2495             and ch$eql (.olduserlen, oldusername,
2415 2496                         .locuser_len, netbuf[naf$t_localuser])
2416 2497             then
2417 2498                 begin
2418 2499                     ch$copy (uaf$s_username, recbuf[uaf$t_username], ' ',
2419 2500                             naf$s_localuser, netbuf[naf$t_localuser]);
2420 2501                     $update (rab = nafrab);
2421 2502                     netuaf_modified = true;
2422 2503                     security_audit (nsa$k_recid_netuaf_mod, oldusername);
2423 2504                     end;
2424 2505                 end;
2425 2506             end;
2426 2507
2427 2508     if .modified then
```

2428  
2429  
2430  
2431  
2432  
2433  
24342509 2 LIB\$SIGNAL(UAF\$\_ZZPRACREN, 2, .olduserlen, oldusername);  
2510  
2511  
2512  
2513  
2514  
2515 1 Return to keyed access  
2515 1 nafrab[ra\$b\_rac] = ra\$b\_key;  
2515 1 end;

				07FC 00000	.EXTRN SYSSREWIND	
					.ENTRY ADJUST_PROXY, Save R2,R3,R4,R5,R6,R7,R8,R9,-	2396
					R10	
					MOVAB SECURITY_AUDIT, R10	
					MOVAB OLDUSERNAME, R9	
					MOVAB NETBUF+64, R8	
					CLRL MODIFIED	2397
	6C	AB	00080000	8F D0 00019	MOVL #524288, NAFRAB+4	2444
			0086	C8 94 00021	CLRB NAFRAB+30	2445
			68	A8 9F 00025	PUSHAB NAFRAB	2446
	00000000G	00		01 FB 00028	CALLS #1, SYSSREWIND	
	00000000V	00		00 FB 0002F	CALLS #0, GET_PROXY_RECORD	2451
		57		50 D0 00036	MOVL R0, STATUS	
		03		57 E8 00039	BLBS STATUS, 2\$	
				0081 31 0003C	BRW 8\$	
68		20		20 3A 0003F	LOCC #32, #32, NETBUF+64	2466
				02 12 00043	BNEQ 3\$	
				51 D4 00045	CLRL R1	
				51 D5 00047	TSTL BLANK_PTR	
				05 12 00049	BNEQ 4\$	
		50		20 D0 0004B	MOVL #32, LOCUSER_LEN	2468
				07 11 0004E	BRB 5\$	
		52		68 9E 00050	MOVAB NETBUF+64, R2	2470
50		51		52 C3 00053	SUBL3 R2, BLANK_PTR, LOCUSER_LEN	
		30	04	AC E9 00057	BLBC REMOVE, 7\$	2478
		51	FA1C	C8 3C 0005B	MOVZWL TOKENLEN, R1	
		50		51 D1 00060	CMPL R1, LOCUSER_LEN	
				CA 12 00063	BNEQ 1\$	
		52	FA20	C8 D0 00065	MOVL TOKENPTR, R2	2479
50		62		51 2D 0006A	CMPC5 R1, (R2), #0, LOCUSER_LEN, NETBUF+64	
				68 0006F		
				BD 12 00070	BNEQ 1\$	
	F4	A9		01 D0 00072	MOVL #1, NETUAF_MODIFIED	2482
			68	A8 9F 00076	PUSHAB NAFRAB	2483
	00000000G	00		01 FB 00079	CALLS #1, SYSSDELETE	
		6A	00020003	8F DD 00080	PUSHL #131075	2484
				01 FB 00086	CALLS #1, SECURITY_AUDIT	
		56		A4 11 00089	BRB 1\$	2475
		51	FC	01 D0 0008B	MOVL #1, MODIFIED	2493
		50		A9 D0 0008E	MOVL OLDUSERLEN, R1	2494
				51 D1 00092	CMPL R1, LOCUSER_LEN	
				98 12 00095	BNEQ 1\$	
50		00		51 2D 00097	CMPC5 R1, OLDUSERNAME, #0, LOCUSER_LEN, NETBUF+64	2496
				68 0009C		
				90 12 0009D	BNEQ 1\$	
	68	FA40	C8	20 28 0009F	MOV C3 #32, RECBUF+4, NETBUF+64	2500

UAFMAIN  
V04-000

adjust\_proxy - implicitly remove/update proxy r

F 1  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 B1ss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 89  
(15)

00000000G	00	68	A8	9F	000A5	PUSHAB	NAFRAB	:	2501
F4	A9		01	FB	000A8	CALLS	#1, SYSSUPDATE	:	
			01	DD	000AF	MOVL	#1, NETUAF_MODIFIED	:	2502
		00030003	59	DD	000B3	PUSHL	R9	:	2503
	6A		8F	DD	000B5	PUSHL	#196611	:	
			02	FB	000BB	CALLS	#2, SECURITY_AUDIT	:	
	14		C9	11	000BE	BRB	6\$	:	2451
			56	E9	000C0	BLBC	MODIFIED, 9\$	:	2508
			59	DD	000C3	PUSHL	R9	:	2509
		FC	A9	DD	000C5	PUSHL	OLDUSERLEN	:	
			02	DD	000C8	PUSHL	#2	:	
		00000000G	8F	DD	000CA	PUSHL	#UAF\$, ZZPRACREN	:	
00000000G	00		04	FB	000D0	CALLS	#4, LIB\$SIGNAL	:	
0086	C8		01	90	000D7	MOVB	#1, NAFRAB+30	:	2514
			04	000DC	RET			:	2515

; Routine Size: 221 bytes, Routine Base: \$CODE\$ + 0D87

default\_uaf - change default record

```
2436 2516 1 %sbttl 'default_uaf - change default record'
2437 2517 1 global routine default_uaf: novalue =
2438 2518 2 begin
2439 2519 2
2440 2520 2 ++
2441 2521 2
2442 2522 2 FUNCTIONAL DESCRIPTION:
2443 2523 2
2444 2524 2     Change a default value in the default record.
2445 2525 2
2446 2526 2 INPUTS:
2447 2527 2
2448 2528 2     none
2449 2529 2
2450 2530 2 IMPLICIT INPUTS:
2451 2531 2
2452 2532 2     none
2453 2533 2
2454 2534 2 OUTPUTS:
2455 2535 2
2456 2536 2     none
2457 2537 2
2458 2538 2 IMPLICIT OUTPUTS:
2459 2539 2
2460 2540 2     none
2461 2541 2
2462 2542 2 ROUTINE VALUE:
2463 2543 2
2464 2544 2     none
2465 2545 2
2466 2546 2 SIDE EFFECTS:
2467 2547 2
2468 2548 2     none
2469 2549 2 --
2470 2550 2
2471 2551 2
2472 2552 2
2473 2553 2 Locate default record and load it into RECBUF
2474 2554 2 if not already there (via an indirect MODIFY DEFAULT)
2475 2555 2
2476 2556 2
2477 2557 2 if not .mod_default
2478 2558 2 then
2479 2559 2     begin
2480 2560 2         if not locate_user (.defuser<0,8>, defuser+1, true)
2481 2561 2         then
2482 2562 2             begin
2483 2563 2                 LIB$SIGNAL(UAF$DEFERR, 0, .rmserr);
2484 2564 2                 return;
2485 2565 2             end;
2486 2566 2         end;
2487 2567 2
2488 2568 2 The encrypted password field of the DEFAULT record can not be propagated
2489 2569 2 to another user, because the encryption algorithm takes the user name as
2490 2570 2 an input. The user is merely warned that this qualifier has no effect.
2491 2571 2
2492 2572 2
```

default\_uaf - change default record

```
2493 2573 2  pwd_flag = true;
2494 2574 2
2495 2575 2  |
2496 2576 2  | Update values supplied and exit if errors.
2497 2577 2  |
2498 2578 2  |
2499 2579 2  if update_record ()
2500 2580 2  then
2501 2581 2  begin
2502 2582 2  if not .pwd_flag
2503 2583 2  then LIB$SIGNAL(UAF$_NODEFPWD);
2504 2584 2  |
2505 2585 2  |
2506 2586 2  | Now write the modified record back into the DEFAULT_RECORD buffer.
2507 2587 2  |
2508 2588 2  default_size = .uafrab[rab$w_rsz];
2509 2589 2  ch$move(.default_size, recbuf, default_record);
2510 2590 2  |
2511 2591 2  |
2512 2592 2  | Update the default record in the file. Note that file has changed.
2513 2593 2  |
2514 2594 2  if not rmsok ($update (rab = uafrab))
2515 2595 2  then
2516 2596 2  LIB$SIGNAL(UAF$_MDFYERR, 0, .rmserr)
2517 2597 2  else
2518 2598 2  begin
2519 2599 2  modify_flag = true;
2520 2600 2  security_audit (nsa$k_recid_sysuaf_mod);
2521 2601 2  if not .mod_default
2522 2602 2  then
2523 2603 2  LIB$SIGNAL(UAF$_MDFYMSG)
2524 2604 2  else
2525 2605 2  mod_default = false;
2526 2606 2  end;
2527 2607 2  end
2528 2608 2
2529 2609 2  else
2530 2610 2  $release (rab = uafrab);
2531 2611 2  ! unlock the record
2532 2612 2  end;
```

```
01FC 00000
5B 00000000G 00 9E 00002
57 00000000' 00 9E 00009
56 00000000' 00 9E 00010
25 67 E8 00017
01 DD 0001A
00 9F 0001C
7E 00000000' 00 9A 00022
00 03 FB 00029
0C 50 E8 00030
66 DD 00033
7E D4 00035
```

```
.ENTRY DEFAULT UAF, Save R2,R3,R4,R5,R6,R7,R8
MOVAB LIB$SIGNAL, R8
MOVAB MOD_DEFAULT, R7
MOVAB RMSERR, R6
BLBS MOD_DEFAULT, 1$
PUSHL #1
PUSHAB DEFUSER+1
MOVZBL DEFUSER, -(SP)
CALLS #3, LOCATE_USER
BLBS R0, 1$
PUSHL RMSERR
CLRL -(SP)
```

2517

2557  
2560

2563



default\_uaf - change default record

1 1  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 92  
(16)

		00000000G	8F	DD	00037	PUSHL	#UAF\$DEFERR		
			48	11	0003D	BRB	3\$		
	0680	C6	01	D0	0003F	1\$:	MOVL	#1, PWD_FLAG	2573
	00000000G	00	00	FB	00044		CALLS	#0, UPDATE_RECORD	2579
		5E	50	E9	0004B		BLBC	R0, 6\$	
		09	C6	E8	0004E		BLBS	PWD_FLAG, 2\$	2582
			8F	DD	00053		PUSHL	#UAF\$NODEFPWD	2583
	0680	68	01	FB	00059		CALLS	#1, LIB\$SIGNAL	
			C6	B0	0005C	2\$:	MOVW	UAFRAB+34, DEFAULT_SIZE	2588
045C	C7	0458	C7	28	00063		MOVC3	DEFAULT_SIZE, RECB0F, DEFAULT_RECORD	2589
		08	A6	C6	9F		PUSHAB	UAFRAB	2594
				01	FB		CALLS	#1, SYS\$UPDATE	
	00000000G	00	50	D0	00077		MOVL	R0, RMSERR	
		66	50	E8	0007A		BLBS	R0, 4\$	
		0E	66	DD	0007D		PUSHL	RMSERR	2596
			7E	D4	0007F		CLRL	-(SP)	
			8F	DD	00081		PUSHL	#UAF\$MDFYERR	
		68	03	FB	00087	3\$:	CALLS	#3, LIB\$SIGNAL	
				04	0008A		RET		
	04	A7	01	D0	0008B	4\$:	MOVL	#1, MODIFY_FLAG	2599
			8F	DD	0008F		PUSHL	#196610	2600
	00000000V	00	01	FB	00095		CALLS	#1, SECURITY_AUDIT	
		0A	67	E8	0009C		BLBS	MOD_DEFAULT, 5\$	2601
			8F	DD	0009F		PUSHL	#UAF\$MDFYMSG	2603
		68	01	FB	000A5		CALLS	#1, LIB\$SIGNAL	
				04	000AB		RET		
			67	D4	000A9	5\$:	CLRL	MOD_DEFAULT	2605
				04	000AB		RET		2579
			C6	9F	000AC	6\$:	PUSHAB	UAFRAB	2610
	00000000G	00	01	FB	000B0		CALLS	#1, SYS\$RELEASE	
				04	000B7		RET		2612

; Routine Size: 184 bytes, Routine Base: \$CODE\$ + 0E64

```
2534 2613 1 %sbttl 'list_proxy - list proxy entries in NETUAF.LIS'
2535 2614 1 global routine list_proxy : novalue =
2536 2615 begin
2537 2616
2538 2617 |++
2539 2618
2540 2619 FUNCTIONAL DESCRIPTION:
2541 2620
2542 2621 This routine produces a listing of the entire NETUAF.DAT file
2543 2622 in NETUAF.LIS.
2544 2623
2545 2624 INPUTS:
2546 2625
2547 2626 none
2548 2627
2549 2628 OUTPUTS:
2550 2629
2551 2630 none
2552 2631
2553 2632 SIDE EFFECTS:
2554 2633
2555 2634 A listing file is produced
2556 2635
2557 2636 |--
2558 2637
2559 2638 local
2560 2639 action;
2561 2640
2562 2641 |
2563 2642 Make sure NETUAF.DAT exists
2564 2643
2565 2644 if not .netuaf exists
2566 2645 then return LIB$SIGNAL(UAF$_NAFDNE);
2567 2646
2568 2647 |
2569 2648 Set up the listing file FAB and connect RAB
2570 2649
2571 2650 nlstfab[fab$_v_dlt] = false;
2572 2651 if rmsbad ($create (fab = nlstfab))
2573 2652 then return LIB$SIGNAL(UAF$_LSTERR, 0, .rmserr);
2574 2653
2575 2654 if rmsbad ($connect (rab = nlstrab))
2576 2655 then return LIB$SIGNAL(UAF$_LSTERR, 0, .rmserr);
2577 2656
2578 2657 |
2579 2658 Set action routine and rab pointer
2580 2659
2581 2660 wild_netuser = true;
2582 2661 match_tokenlen = 1;
2583 2662 match_token = %c'a';
2584 2663 rabptr = nlstrab;
2585 2664 action = display_proxy;
2586 2665 header_flag = true;
2587 2666 nafcab[rab$_l_rop] = rab$_rrl or rab$_nlk;
2588 2667
2589 2668 LIB$SIGNAL(UAF$_LSTMSG1);
2590 2669
```

```

2591      2670      2671      2672      2673      2674      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2592      2671      2672      2673      2674      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2593      2672      2673      2674      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2594      2673      2674      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2595      2674      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2596      2675      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2597      2676      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2598      2677      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2599      2678      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2600      2679      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2601      2680      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2602      2681      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2603      2682      2683      2684      2685      2686      2687      2688      2689      2690      2691
2604      2683      2684      2685      2686      2687      2688      2689      2690      2691
2605      2684      2685      2686      2687      2688      2689      2690      2691
2606      2685      2686      2687      2688      2689      2690      2691
2607      2686      2687      2688      2689      2690      2691
2608      2687      2688      2689      2690      2691
2609      2688      2689      2690      2691
2610      2689      2690      2691
2611      2690      2691
2612      2691

```

```
.EXTRN  SYS$DISCONNECT, SYS$CLOSE
```

Address	Disassembly	Comment	Hex
007C	00000	ENTRY LIST PROXY, Save R2,R3,R4,R5,R6	2614
56	00000000G	MOVL #UAF\$ LSTERR, R6	
55	00000000G	MOVAB LIB\$SIGNAL, R5	
54	00000000'	MOVAB NLSTRAB, R4	
53	00000000'	MOVAB RMSERR, R3	
0A	F8	BLBS NETUAF EXISTS, 1\$	2644
	00000000G	PUSHL #UAF\$ RAFDNE	2645
65		CALLS #1, LIB\$SIGNAL	
		RET	
B5	A4	1\$: BICB2 #128, NLSTFAB+5	2650
	80	PUSHAB NLSTFAB	2651
	00	CALLS #1, SYSS\$CREATE	
00000000G	63	MOVL R0, RMSERR	
	0F	BLBC R0, 2\$	
		PUSHL R4	2654
00000000G	00	CALLS #1, SYSS\$CONNECT	
	63	MOVL R0, RMSERR	
	0A	BLBS R0, 3\$	
		2\$: PUSHL RMSERR	2655
		CLRL -(SP)	
		PUSHL R6	
65		CALLS #3, LIB\$SIGNAL	
		RET	
E0	A3	3\$: MOVL #1, WILD NETUSER	2660
DC	A3	MOVL #1, MATCH TOKENLEN	2661
98	A3	MOVL #42, MATCH TOKEN	2662
80	A3	MOVAB NLSTRAB, RABPTR	2663
	52	MOVAB DISPLAY PROXY, ACTION	2664
FCC4	C4	MOVL #1, HEADER FLAG	2665
0638	C3	MOVL #1048584, RABRAB+4	2666

UAFMAIN  
V04-000

list\_proxy - list proxy entries in NETUAF.LIS

L 1  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 95  
(17)

		00000000G	8F	DD	0007F	PUSHL	#UAF\$ LSTMSG1	2668
	65		01	FB	00085	CALLS	#1, LIB\$SIGNAL	
			52	DD	00088	PUSHL	ACTION	2673
00000000V	00		01	FB	0008A	CALLS	#1, LOCATE_PROXY	
	63		50	DD	00091	MOVL	R0, RMSERR	
	1F		50	E8	00094	BLBS	R0, 5\$	
	50		63	DD	00097	MOVL	RMSERR, R0	2676
000182B2	8F		50	D1	0009A	CMPL	R0, #98994	
			08	12	000A1	BNEQ	4\$	
		00000000G	8F	DD	000A3	PUSHL	#UAF\$_BADSPC	2678
			11	11	000A9	BRB	6\$	
			50	DD	000AB	PUSHL	R0	2680
			7E	D4	000AD	CLRL	-(SP)	
	65		56	DD	000AF	PUSHL	R6	
			03	FB	000B1	CALLS	#3, LIB\$SIGNAL	2673
			09	11	000B4	BRB	7\$	2683
		00000000G	8F	DD	000B6	PUSHL	#UAF\$ NETLSTMSG	
	65		01	FB	000BC	CALLS	#1, LIB\$SIGNAL	2688
			54	DD	000BF	PUSHL	R4	
00000000G	00		01	FB	000C1	CALLS	#1, SYS\$DISCONNECT	2689
		B0	A4	9F	000C8	PUSHAB	NLSTFAB	
00000000G	00		01	FB	000CB	CALLS	#1, SYS\$CLOSE	2691
			04	000D2	RET			

; Routine Size: 211 bytes, Routine Base: \$CODE\$ + 0F1C

## list\_uaf - List file routine

```
2614 2692 1 %sbttl 'list_uaf - list file routine'
2615 2693 1 global routine list_uaf : novalue =
2616 2694 2 begin
2617 2695 3
2618 2696 4 ++
2619 2697 5
2620 2698 6 FUNCTIONAL DESCRIPTION:
2621 2699 7
2622 2700 8     Display the specified users in a file named 'SYSUAF.LIS'.
2623 2701 9
2624 2702 10 INPUTS:
2625 2703 11
2626 2704 12     none
2627 2705 13
2628 2706 14 IMPLICIT INPUTS:
2629 2707 15
2630 2708 16     none
2631 2709 17
2632 2710 18 OUTPUTS:
2633 2711 19
2634 2712 20     none
2635 2713 21
2636 2714 22 IMPLICIT OUTPUTS:
2637 2715 23
2638 2716 24     none
2639 2717 25
2640 2718 26 ROUTINE VALUE:
2641 2719 27
2642 2720 28     none
2643 2721 29
2644 2722 30 SIDE EFFECTS:
2645 2723 31
2646 2724 32     none
2647 2725 33 --
2648 2726 34
2649 2727 35 local
2650 2728 36     action;
2651 2729 37
2652 2730 38
2653 2731 39 Obtain the user specification. This sets wildcard flags and initializes
2654 2732 40 the appropriate key in RECBUF.
2655 2733 41
2656 2734 42
2657 2735 43 if not parse_wild (sd_token1,true)          ! Null string defaults to *
2658 2736 44 then return;
2659 2737 45
2660 2738 46
2661 2739 47 Obtain qualifiers. This determines which display should be used.
2662 2740 48
2663 2741 49 full_flag = false;
2664 2742 50 brief_flag = true;
2665 2743 51
2666 2744 52 if cli$present (sd_full) or (not cli$present (sd_brief))
2667 2745 53 then
2668 2746 54     begin
2669 2747 55         brief_flag = false;
2670 2748 56         full_flag = true;
```



## list\_uaf - list file routine

```
2671 2749      end;
2672 2750
2673 2751      |
2674 2752      | Create the listing file.
2675 2753      |
2676 2754      |
2677 2755      | lsfab[fab$y_dlt] = false;          ! initialize DLT bit
2678 2756      | if rmsbad ($create (fab = lsfab))
2679 2757      | then return LIB$SIGNAL(UAF$_LSTERR, 0, .rmserr);
2680 2758
2681 2759      | if rmsbad ($connect (rab = lstrab))
2682 2760      | then return LIB$SIGNAL(UAF$_LSTERR, 0, .rmserr);
2683 2761      |
2684 2762      |
2685 2763      | Request a header record for the file and aim RABPTR at our RAB.
2686 2764      |
2687 2765      |
2688 2766      | header_flag = true;
2689 2767      | rabptr = lstrab;
2690 2768      | found_match = false;
2691 2769      | uafra[ra$b_l_rop] = ra$b_m_rrl or ra$b_m_nlk;
2692 2770      |
2693 2771      | Flag the list operation for the rights data base routines .
2694 2772      |
2695 2773      | rdb_list_flag = true ;
2696 2774      |
2697 2775      |
2698 2776      | Choose the appropriate display.
2699 2777      |
2700 2778      |
2701 2779      | action = display_brief;
2702 2780      | if .full_flag
2703 2781      | then action = display_full;
2704 2782      |
2705 2783      | LIB$SIGNAL(UAF$_LSTMSG1);          ! announce starting
2706 2784      |
2707 2785      | if rmsbad (wild_user (.action))
2708 2786      | then
2709 2787      |     begin
2710 2788      |         if .rmserr eql rms$_rnf
2711 2789      |         then
2712 2790      |             LIB$SIGNAL(UAF$_BADSPC)
2713 2791      |         else
2714 2792      |             LIB$SIGNAL(UAF$_LSTERR, 0, .rmserr);
2715 2793      |             lsfab[fab$y_dlt] = true;          ! press delete button
2716 2794      |         end
2717 2795      |     else
2718 2796      |         LIB$SIGNAL(UAF$_LSTMSG2);
2719 2797      |
2720 2798      | $disconnect (rab = lstrab);
2721 2799      | $close (fab = lsfab);
2722 2800      | end;
```

			01FC 00000	.ENTRY	LIST UAF, Save R2,R3,R4,R5,R6,R7,R8	2693
	58	00000000G	8F D0 00002	MOVL	#UAF\$ LSTERR, R8	
	57	00000000G	00 9E 00009	MOVAB	CLISPRESENT, R7	
	56	00000000G	00 9E 00010	MOVAB	SD TOKEN1, R6	
	55	00000000G	00 9E 00017	MOVAB	LIB\$SIGNAL, R5	
	54	00000000G	00 9E 0001E	MOVAB	RMSERR, R4	
	53	00000000G	00 9E 00025	MOVAB	LSTRAB, R3	
			01 DD 0002C	PUSHL	#1	2735
			56 DD 0002E	PUSHL	R6	
00000000G	00		02 FB 00030	CALLS	#2, PARSE_WILD	
	01		50 E8 00037	BLBS	R0, 1\$	
			04 0003A	RET		
FE08	C3		01 7D 0003B 1\$:	MOVQ	#1, BRIEF_FLAG	2742
		2C	A6 9F 00040	PUSHAB	SD_FULL	2744
	67		01 FB 00043	CALLS	#1, CLISPRESENT	
	09		50 E8 00046	BLBS	R0, 2\$	
		20	A6 9F 00049	PUSHAB	SD_BRIEF	
	67		01 FB 0004C	CALLS	#1, CLISPRESENT	
	09		50 E8 0004F	BLBS	R0, 3\$	
		FE08	C3 D4 00052 2\$:	CLRL	BRIEF_FLAG	2747
FE0C	C3		01 D0 00056	MOVL	#1, FOLL_FLAG	2748
B5	A3	80	8F 8A 0005B 3\$:	BICB2	#128, LSTFAB+5	2755
		B0	A3 9F 00060	PUSHAB	LSTFAB	2756
00000000G	00		01 FB 00063	CALLS	#1, SYSS\$CREATE	
	64		50 D0 0006A	MOVL	R0, RMSERR	
	0F		50 E9 0006D	BLBC	R0, 4\$	
			53 DD 00070	PUSHL	R3	2759
00000000G	00		01 FB 00072	CALLS	#1, SYSS\$CONNECT	
	64		50 D0 00079	MOVL	R0, RMSERR	
	0A		50 E8 0007C	BLBS	R0, 5\$	
			64 DD 0007F 4\$:	PUSHL	RMSERR	2760
			7E D4 00081	CLRL	-(SP)	
			58 DD 00083	PUSHL	R8	
	65		03 FB 00085	CALLS	#3, LIB\$SIGNAL	
			04 00088	RET		
FE10	C3		01 D0 00089 5\$:	MOVL	#1, HEADER_FLAG	2766
80	A4		63 9E 0008E	MOVAB	LSTRAB, RABPTR	2767
		06C8	C4 D4 00092	CLRL	FOUND_MATCH	2768
05F4	C4	00100008	8F D0 00096	MOVL	#1048584, UAFRAB+4	2769
00000000G	00		01 90 0009F	MOVB	#1, RDB_LIST_FLAG	2773
	52	00000000V	00 9E 000A6	MOVAB	DISPLAY-BRIEF, ACTION	2779
	07	FE0C	C3 E9 000AD	BLBC	FULL_FLAG, 6\$	2780
	52	00000000V	00 9E 000B2	MOVAB	DISPCAY FULL, ACTION	2781
		00000000G	8F DD 000B9 6\$:	PUSHL	#UAF\$ LSTMSG1	2783
	65		01 FB 000BF	CALLS	#1, LIB\$SIGNAL	
			52 DD 000C2	PUSHL	ACTION	2785
00000000V	00		01 FB 000C4	CALLS	#1, WILD_USER	
	64		50 D0 000CB	MOVL	R0, RMSERR	
	27		50 E8 000CE	BLBS	R0, 9\$	
	50		64 D0 000D1	MOVL	RMSERR, R0	2788
000182B2	8F		50 D1 000D4	CMPL	R0, #98994	
			0B 12 000DB	BNEQ	7\$	
		00000000G	8F DD 000DD	PUSHL	#UAF\$ BADSPC	2790
	65		01 FB 000E3	CALLS	#1, LIB\$SIGNAL	
			09 11 000E6	BRB	8\$	
			50 DD 000E8 7\$:	PUSHL	R0	2792
			7E D4 000EA	CLRL	-(SP)	

UAFMAIN  
V04-000

list\_uaf - list file routine

C 2  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 BLISS-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 99  
(18)

			58	DD	000EC		PUSHL	R8		
			03	FB	000EE		CALLS	#3,	LIBSSIGNAL	
B5	65		8F	88	000F1	8\$:	BISB2	#128,	LSTFAB+5	2793
	A3	80	09	11	000F6		BRB	10\$		2785
		00000000G	8F	DD	000F8	9\$:	PUSHL	#UAF\$,	LSTMSG2	2796
	65		01	FB	000FE		CALLS	#1,	LIBSSIGNAL	
00000000G	00		53	DD	00101	10\$:	PUSHL	R3		2798
			01	FB	00103		CALLS	#1,	SYSSDISCONNECT	
		B0	A3	9F	0010A		PUSHAB	LSTFAB		2799
00000000G	00		01	FB	0010D		CALLS	#1,	SYSSCLOSE	
			04	00114			RET			2800

; Routine Size: 277 bytes, Routine Base: \$CODE\$ + 0FEF

UA  
VO

show\_user\_uaf - display user record

```
2724 2801 1 %sbttl 'show_user_uaf - display user record'
2725 2802 1 global routine show_user_uaf : novalue =
2726 2803 1 begin
2727 2804 1
2728 2805 1 ++
2729 2806 1
2730 2807 1 FUNCTIONAL DESCRIPTION:
2731 2808 1
2732 2809 1     Display the specified users on SYS$OUTPUT.
2733 2810 1
2734 2811 1 INPUTS:
2735 2812 1
2736 2813 1     none
2737 2814 1
2738 2815 1 IMPLICIT INPUTS:
2739 2816 1
2740 2817 1     none
2741 2818 1
2742 2819 1 OUTPUTS:
2743 2820 1
2744 2821 1     none
2745 2822 1
2746 2823 1 IMPLICIT OUTPUTS:
2747 2824 1
2748 2825 1     none
2749 2826 1
2750 2827 1 ROUTINE VALUE:
2751 2828 1
2752 2829 1     none
2753 2830 1
2754 2831 1 SIDE EFFECTS:
2755 2832 1
2756 2833 1     none
2757 2834 1 --
2758 2835 1
2759 2836 1 local
2760 2837 1     action;
2761 2838 1
2762 2839 1
2763 2840 1 Obtain the user specification. This sets wildcard flags and initializes
2764 2841 1 the appropriate key in RECBUF.
2765 2842 1
2766 2843 1
2767 2844 1 if not parse_wild (sd_token1,false)          ! Null string is disallowed.
2768 2845 1 then return;
2769 2846 1
2770 2847 1
2771 2848 1 Obtain qualifiers. This determines which display should be used.
2772 2849 1
2773 2850 1 full_flag = true;
2774 2851 1 brief_flag = false;
2775 2852 1
2776 2853 1 if cli$present (sd_brief) or (not cli$present (sd_full))
2777 2854 1 then
2778 2855 1     begin
2779 2856 1         brief_flag = true;
2780 2857 1         full_flag = false;
```

```

2781 2858      end;
2782 2859
2783 2860      |
2784 2861      | Request a header record for the file and aim RABPTR at our RAB.
2785 2862      |
2786 2863      |
2787 2864      | header_flag = true;
2788 2865      | rabptr = outrab;
2789 2866      | found_match = false;
2790 2867      | uafra6[rab$l_rop] = rab$m_rri or rab$m_nlk;
2791 2868      |
2792 2869      |
2793 2870      | Flag the show operation for the rights data base routines .
2794 2871      |
2795 2872      | rdb_list_flag = false ;
2796 2873      |
2797 2874      |
2798 2875      | Choose the appropriate display.
2799 2876      |
2800 2877      |
2801 2878      | action = display_full;
2802 2879      | if .brief_flag
2803 2880      | then action = display_brief;
2804 2881      |
2805 2882      | if rmsbad (wild_user (.action))
2806 2883      | then
2807 2884      |     begin
2808 2885      |         if .rmserr eql rms$_rnf
2809 2886      |         then
2810 2887      |             LIB$SIGNAL(UAF$_BADSPC)
2811 2888      |         else
2812 2889      |             LIB$SIGNAL(UAF$_SHOW_ERR, 0, .rmserr);
2813 2890      |         end;
2814 2891      | end;

```

```

00000000G 00 007C 00000
00000000G 00 9E 00002
00000000G 00 9E 00009
00000000' 00 9E 00010
00000000' 00 9E 00017
00000000' 00 9E 0001E
00000000' 7E D4 00025
00000000G 00 54 DD 00027
00000000G 00 02 FB 00029
00000000G 04 A3 50 E9 00030
00000000G 00 01 D0 00033
00000000G 20 63 D4 00037
00000000G 00 A4 9F 00039
00000000G 09 01 FB 0003C
00000000G 2C 50 E8 0003F
00000000G 00 A4 9F 00042
00000000G 03 01 FB 00045
00000000G 00 50 E8 00048

```

```

.ENTRY SHOW USER UAF, Save R2,R3,R4,R5,R6
MOVAB LIB$SIGNAL, R6
MOVAB CL$PRESENT, R5
MOVAB SD_TOKEN1, R4
MOVAB BRIEF_FLAG, R3
MOVAB RMSERR, R2
CLRL -(SP)
PUSHL R4
CALLS #2, PARSE_WILD
BLBC R0, 5$
MOVL #1, FULL_FLAG
CLRL BRIEF_FLAG
PUSHAB SD_BRIEF
CALLS #1, CL$PRESENT
BLBS R0, 1$
PUSHAB SD_FULL
CALLS #1, CL$PRESENT
BLBS R0, 2$

```

2802

2844

2850

2851

2853



## show\_user\_uaf - display user record

F 2  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 102  
(19)

08	63	01	7D	0004B	1\$:	MOVQ	#1, BRIEF FLAG	2856
80	A3	01	D0	0004E	2\$:	MOVL	#1, HEADER FLAG	2864
	A2	C2	9E	00052		MOVAB	OUTRAB, RABPTR	2865
	0684	C2	D4	00058		CLRL	FOUND MATCH	2866
05F4	C2	8F	D0	0005C		MOVL	#1048584, UAFRAB+4	2867
	00000000G	00	94	00065		CLRB	RDB LIST FLAG	2872
	50	00	9E	0006B		MOVAB	DISPLAY FULL ACTION	2878
	07	63	E9	00072		BLBC	BRIEF FLAG, 3\$	2879
	50	00	9E	00075		MOVAB	DISPLAY BRIEF ACTION	2880
		50	DD	0007C	3\$:	PUSHL	ACTION	2882
00000000V	00	01	FB	0007E		CALLS	#1, WILD USER	
	62	50	D0	00085		MOVL	R0, RMSERR	
	23	50	E8	00088		BLBS	R0, 5\$	
	50	62	D0	0008B		MOVL	RMSERR, R0	2885
000182B2	8F	50	D1	0008E		CMPL	R0, #98994	
		0A	12	00095		BNEQ	4\$	
	00000000G	8F	DD	00097		PUSHL	#UAF\$ BADSPC	2887
	66	01	FB	0009D		CALLS	#1, LIB\$ SIGNAL	
			04	000A0		RET		
		50	DD	000A1	4\$:	PUSHL	R0	2889
		7E	D4	000A3		CLRL	-(SP)	
	00000000G	8F	DD	000A5		PUSHL	#UAF\$ SHOW ERR	
	66	03	FB	000AB		CALLS	#3, LIB\$ SIGNAL	
		04	000AE	5\$:	RET			2891

; Routine Size: 175 bytes. Routine Base: \$CODE\$ + 1104

```
2816 2892 1 %sbttl 'show_proxy - display proxy record at terminal'
2817 2893 1 global routine show_proxy : novalue =
2818 2894 1 begin
2819 2895 1
2820 2896 1 ++
2821 2897 1
2822 2898 1 FUNCTIONAL DESCRIPTION:
2823 2899 1
2824 2900 1     This routine will display a specific proxy record or
2825 2901 1     will display all proxy entries to the user terminal.
2826 2902 1
2827 2903 1 INPUTS:
2828 2904 1
2829 2905 1     none
2830 2906 1
2831 2907 1 OUTPUTS:
2832 2908 1
2833 2909 1     none
2834 2910 1
2835 2911 1 IMPLICIT INPUTS:
2836 2912 1
2837 2913 1     TOKENLEN, TOKENPTR
2838 2914 1
2839 2915 1 IMPLICIT OUTPUTS:
2840 2916 1
2841 2917 1     none
2842 2918 1
2843 2919 1 SIDE EFFECTS:
2844 2920 1
2845 2921 1     none
2846 2922 1
2847 2923 1 --
2848 2924 1
2849 2925 1 local
2850 2926 1     node_len,
2851 2927 1     node_ptr,
2852 2928 1     remuser_len,
2853 2929 1     remuser_ptr,
2854 2930 1     action,
2855 2931 1     counter,
2856 2932 1     success;
2857 2933 1
2858 2934 1
2859 2935 1     Make sure that NETUAF.DAT exists
2860 2936 1
2861 2937 1     if not .netuaf exists
2862 2938 1     then return LIB$SIGNAL(UAF$_NAFDNE);
2863 2939 1
2864 2940 1
2865 2941 1     Retrieve token
2866 2942 1
2867 2943 1     cli$get_value (sd_token1, tokendsc);
2868 2944 1
2869 2945 1     header_flag = true;
2870 2946 1
2871 2947 1
2872 2948 1     Wildcard spec?
```

```
2873 2949 2 |
2874 2950 2 | if ch$find_ch (.tokenlen, .tokenptr, %c'%') neq 0
2875 2951 2 | or ch$find_ch (.tokenlen, .tokenptr, %c'+'') neq 0
2876 2952 2 | then
2877 2953 2 |     begin
2878 2954 2 |         wild_netuser = true;
2879 2955 2 |         match_tokenlen = .tokenlen;
2880 2956 2 |         ch$move (.tokenlen, .tokenptr, match_token);
2881 2957 2 |         end
2882 2958 2 |
2883 2959 2 |         Otherwise, just display a single entry
2884 2960 2 |
2885 2961 2 |     else
2886 2962 2 |         begin
2887 2963 2 |             wild_netuser = false;
2888 2964 2 |             if not remote_parse (node_ptr, node_len, remuser_ptr, remuser_len)
2889 2965 2 |                 then return;
2890 2966 2 |
2891 2967 2 |             ch$copy (.node_len, .node_ptr, ' ', naf$s_node, netbuf[naf$st_node]);
2892 2968 2 |             ch$copy (.remuser_len, .remuser_ptr, ' ', naf$s_remuser, netbuf[naf$st_remuser]);
2893 2969 2 |             naf$s_remuser, netbuf[naf$st_remuser]);
2894 2970 2 |         end;
2895 2971 2 |
2896 2972 2 |         Set up action routine and rab pointer
2897 2973 2 |
2898 2974 2 |         rabptr = outrab;
2899 2975 2 |         action = display_proxy;
2900 2976 2 |         found_match = false;
2901 2977 2 |         naf$rab[rab$st_rop] = rab$m_rri or rab$m_nlk;
2902 2978 2 |
2903 2979 2 |         Make call (s) necessary to display the requested entry or entries
2904 2980 2 |
2905 2981 2 |         if rmsbad (locate_proxy (.action))
2906 2982 2 |         then
2907 2983 2 |             begin
2908 2984 2 |                 if .rmserr eql rms$_rnf
2909 2985 2 |                 then
2910 2986 2 |                     LIB$SIGNAL(UAF$_BADSPC)
2911 2987 2 |                 else
2912 2988 2 |                     LIB$SIGNAL(UAF$_SHOW_ERR, 0, .rmserr);
2913 2989 2 |                 end;
2914 2990 2 |             end;
2915 2991 2 |         end;
2916 2992 2 | end;
```

```
00FC 00000
57 00000000G 00 9E 00002
56 00000000' 00 9E 00009
5E 10 C2 00010
09 10 A6 E8 00013
00000000G 8F DD 00017
00AA 31 0001D
56 DD 00020 1$:
```

```
.ENTRY SHOW_PROXY, Save R2,R3,R4,R5,R6,R7
MOVAB LIB$SIGNAL, R7
MOVAB TOKENDSC, R6
SUBL2 #16, SP
BLBS NETUAF_EXISTS, 1$
PUSHL #UAF$_RAFDNE
BRW 7$
PUSHL R6
```

```
: 2893
:
: 2937
: 2938
: 2943
```

			00000000G	00	00000000'	00	9F	00022	PUSHAB	SD_TOKEN1		
			00000000'	00		02	FB	00028	CALLS	#2, CLISGET VALUE		
				52		01	D0	0002F	MOVL	#1, HEADER_FLAG		2945
				53		66	3C	00036	MOVZWL	TOKENLEN, R2		2950
				53		A6	D0	00039	MOVL	TOKENPTR, R3		
				52	04	25	3A	00030	LOCC	#37, R2, (R3)		
						02	12	00041	BNEQ	2\$		
						51	D4	00043	CLRL	R1		
						51	D5	00045	TSTL	R1		
						OC	12	00047	BNEQ	4\$		
						2A	3A	00049	LOCC	#42, R2, (R3)		2951
						02	12	0004D	BNEQ	3\$		
						51	D4	0004F	CLRL	R1		
						51	D5	00051	TSTL	R1		
						OF	13	00053	BEQL	5\$		
			F8	A6		01	D0	00055	MOVL	#1, WILD_NETUSER		2954
			F4	A6		52	D0	00059	MOVL	R2, MATCH_TOKENLEN		2955
				63		52	28	0005D	MOVC3	R2, (R3), MATCH_TOKEN		2956
						29	11	00062	BRB	6\$		2950
						F8	A6	D4	00064	CLRL	WILD_NETUSER	2963
						5E	DD	00067	PUSHL	SP		2964
						08	AE	9F	00069	PUSHAB	REMUSER_PTR	
						10	AE	9F	0006C	PUSHAB	NODE_LEN	
						18	AE	9F	0006F	PUSHAB	NODE_PTR	
			F412	CF		04	FB	00072	CALLS	#4, REMOTE_PARSE		
				61		50	E9	00077	BLBC	R0, 9\$		
20	20		OC	BE		08	AE	2C	0007A	MOVC5	NODE_LEN, @NODE_PTR, #32, #32, NETBUF	2967
						05A4	C6	00081				
20	20		04	BE		6E	2C	00084	MOVC5	REMUSER_LEN, @REMUSER_PTR, #32, #32, -		2969
						C6		0008A		NETBUF+32		
			98	A6		C6	9E	0008D	MOVAB	OUTRAB, RABPTR		2975
				50	00000000V	00	9E	00093	MOVAB	DISPLAY_PROXY, ACTION		2976
						C6	D4	0009A	CLRL	FOUND_MATCH		2977
						8F	D0	0009E	MOVL	#1048584, NAFRAB+4		2978
			0650	C6	00100008	50	DD	000A7	PUSHL	ACTION		2983
				00		01	FB	000A9	CALLS	#1, LOCATE_PROXY		
			00000000V	A6		50	D0	000B0	MOVL	R0, RMSERR		
				24		50	E8	000B4	BLBS	R0, 9\$		
				50	18	A6	D0	000B7	MOVL	RMSERR, R0		2986
			000182B2	8F		50	D1	000BB	CMPL	R0, #98994		
						0A	12	000C2	BNEQ	8\$		
						8F	DD	000C4	PUSHL	#UAF\$ BADSPC		2988
				67	00000000G	01	FB	000CA	CALLS	#1, LIB\$SIGNAL		
						04	000CD		REI			
						50	DD	000CE	PUSHL	R0		2990
						7E	D4	000D0	CLRL	-(SP)		
						8F	DD	000D2	PUSHL	#UAF\$ SHOW_ERR		
				67	00000000G	03	FB	000D8	CALLS	#3, LIB\$SIGNAL		
						04	000DB	9\$:	RET			2992

; Routine Size: 220 bytes, Routine Base: \$CODE\$ + 11B3

```
2918 2993 1 %sbtll 'locate_proxy - access given proxy record (s)'
2919 2994 1 routine locate_proxy (action) =
2920 2995 1 begin
2921 2996 1
2922 2997 1 ++
2923 2998 1
2924 2999 1 FUNCTIONAL DESCRIPTION:
2925 3000 1
2926 3001 1 This routine will call a requested action routine a number of times.
2927 3002 1
2928 3003 1 INPUTS:
2929 3004 1
2930 3005 1 ACTION - the action routine to call for each NEUAF record
2931 3006 1
2932 3007 1 OUTPUTS:
2933 3008 1
2934 3009 1 none
2935 3010 1
2936 3011 1 SIDE EFFECTS:
2937 3012 1
2938 3013 1 none
2939 3014 1
2940 3015 1 --
2941 3016 1 local
2942 3017 1 status;
2943 3018 1
2944 3019 1
2945 3020 1
2946 3021 1 If wild user, set acces to sequential and fetch all records
2947 3022 1
2948 3023 1 if .wild_netuser
2949 3024 1 then
2950 3025 1 begin
2951 3026 1 nafab[raab$rac] = raab$seq;
2952 3027 1 $rewind (rab = nafab);
2953 3028 1 end;
2954 3029 1
2955 3030 1 status = get_proxy_record ();
2956 3031 1
2957 3032 1
2958 3033 1 Fetch record and call action routine until EOF
2959 3034 1
2960 3035 1 if .status
2961 3036 1 then (.action) ();
2962 3037 1
2963 3038 1 if .wild_netuser
2964 3039 1 then
2965 3040 1 begin
2966 3041 1 while status = get_proxy_record ()
2967 3042 1 do (.action) ();
2968 3043 1 if .status eql rms$eof
2969 3044 1 then status = true;
2970 3045 1 end;
2971 3046 1
2972 3047 1
2973 3048 1 Restore keyed access
2974 3049 1
```



```

: 2975
: 2976
: 2977
: 2978
: 2979
: 2980
: 2981
3050 2 nafra[ra$b_rac] = ra$b_key;
3051
3052 if .wild netuser and not .found_match
3053 then LIB$SIGNAL(UAF$_BADSPC);
3054
3055 .status
3056 1 end;

```

001C 00000 LOCATE_PROXY:						
	54	00000000V	00	9E	00002	.WORD
	53	00000000V	00	9E	00009	MOVAB
	OF		63	E9	00010	MOVAB
		0672	C3	94	00013	BLBC
		0654	C3	9F	00017	CLRAB
						NAFRAB+30
00000000G	00		01	FB	0001B	PUSHAB
	64		00	FB	00022	CALLS
	52		50	D0	00025	#1, SYSSREWIND
	04		52	E9	00028	#0, GET_PROXY_RECORD
04	BC		00	FB	00028	MOVAB
	1B		63	E9	0002F	BLBC
	64		00	FB	00032	CALLS
	52		50	D0	00035	#0, GET_PROXY_RECORD
	06		52	E9	00038	MOVAB
04	BC		00	FB	00038	BLBC
			F1	11	0003F	CALLS
0001827A	8F		52	D1	00041	#0, ACTION
			03	12	00048	BRB
	52		01	D0	0004A	3\$
0672	C3		01	90	0004D	CMPL
	12		63	E9	00052	BNEQ
	0D	06E8	C3	E8	00055	5\$
		00000000G	8F	DD	0005A	MOVAB
00000000G	00		01	FB	00060	#1, STATUS
	50		52	D0	00067	#1, NAFRAB+30
			04	0006A		BLBC
						WILD NETUSER, 6\$
						BLBS
						FOUND_MATCH, 6\$
						PUSHL
						#UAF\$_BADSPC
						CALLS
						#1, LIB\$SIGNAL
						MOVAB
						STATUS, R0
						RET

; Routine Size: 107 bytes, Routine Base: \$CODE\$ + 128F

```
2983 3057 1 %sbttl 'get_proxy_record - read single proxy record'
2984 3058 1 routine get_proxy_record =
2985 3059 2 begin
2986 3060 2 ++
2987 3061 2
2988 3062 2
2989 3063 2 FUNCTIONAL DESCRIPTION:
2990 3064 2
2991 3065 2 This routine accesses a specific NETUAF.DAT record
2992 3066 2
2993 3067 2 INPUTS:
2994 3068 2
2995 3069 2 none
2996 3070 2
2997 3071 2 OUTPUTS:
2998 3072 2
2999 3073 2 none
3000 3074 2
3001 3075 2 SIDE EFFECTS:
3002 3076 2
3003 3077 2 none
3004 3078 2
3005 3079 2 --
3006 3080 2
3007 3081 2 local
3008 3082 2 counter,
3009 3083 2 success;
3010 3084 2
3011 3085 2 counter = retry_rlk;
3012 3086 2
3013 3087 2 while ((success = $get (rab = nafrab)) eql rms$_rlk)
3014 3088 2 and ((counter = .counter - 1) geq 0)
3015 3089 2 do
3016 3090 2 if $schdwk (daytim = wakedelta) then $hiber;
3017 3091 2
3018 3092 2 .success
3019 3093 2 end;
```

```
.EXTRN SYSSGET, SYSSCHDWK
.EXTRN SYSSHIBER
```

## 000C 00000 GET\_PROXY\_RECORD:

	52		08	D0	00002	1%:	WORD	Save R2,R3
		00000000'	00	9F	00005		MOVL	#8, COUNTER
			01	FB	00008		PUSHAB	NAFRAB
00000000G	00		50	D0	00012		CALLS	#1, SYSSGET
	53		53	D1	00015		MOVL	R0, SUCCESS
000182AA	8F		21	12	0001C		CMPL	SUCCESS, #98986
			52	D7	0001E		BNEQ	2\$
			1D	19	00020		DECL	COUNTER
			7E	D4	00022		BLSS	2\$
		00000000'	00	9F	00024		CLRL	-(SP)
			7E	7C	0002A		PUSHAB	WAKEDELTA
00000000G	00		04	FB	0002C		CLRG	-(SP)
							CALLS	#4, SYSSCHDWK

```
3058
3085
3087
3088
3090
```

UAFMAIN  
V04-000

get\_proxy\_record - read single proxy record

M 2  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 109  
(22)

00000000G	CF	50	E9	00033	BLBC	R0, 1\$
	00	00	FB	00036	CALLS	#0, SYS\$HIBER
		C6	11	0003D	BRB	1\$
	50	53	D0	0003F	MOVL	SUCCESS, R0
			04	00042	RET	

3093

; Routine Size: 67 bytes, Routine Base: \$CODE\$ + 12FA

```
display_proxy - format and output a proxy entry

3094 1 %sbttl 'display_proxy - format and output a proxy entry'
3095 routine display_proxy : novalue =
3096 begin
3097
3098 ++
3099
3100 FUNCTIONAL DESCRIPTION:
3101
3102     This routine formats and outputs a line of a NETUAF.DAT entry
3103
3104 INPUTS:
3105
3106     none
3107
3108 OUTPUTS:
3109
3110     none
3111
3112 SIDE EFFECTS:
3113
3114     none
3115
3116 --
3117
3118 bind
3119     nafhdr          = cstring (' Node   Remote User   Local User'),
3120     shownaf         = cstring ('!6AD::!12AD   !12AD');
3121
3122 if .wild_netuser
3123 and not
3124 begin
3125     local
3126     nodelen, usrlen, proxy_buf : vector[naf$s_remname+2,byte];
3127     map
3128     dbl_colon : vector;
3129
3130     nodelen = namelen (naf$s_node, netbuf[naf$t_node]);
3131     usrlen  = namelen (naf$s_remuser, netbuf[naf$t_remuser]);
3132     ch$move (.nodelen, netbuf[naf$t_node], proxy_buf [0]);
3133     ch$move (2, .dbl_colon[1], proxy_buf [.nodelen]);
3134     ch$move (.usrlen, netbuf [naf$t_remuser], proxy_buf [.nodelen+2]);
3135
3136     usrlen = .nodelen + .usrlen + 2;
3137     fmg$match_name (.usrlen, proxy_buf, .match_tokenlen, match_token)
3138 end
3139 then
3140     return;
3141
3142 found_match = true;
3143
3144 if .header_flag
3145 then
3146     begin
3147         faomac (nafhdr);
3148         output_null;
3149         header_flag = false;
3150
```

```
3078      3151      2      end;
3079      3152      2
3080      3153      2      faomac (shownaf,
3081      3154      2      ***      naf$s_node,      netbuf[naf$t_node],
3082      3155      2      ***      naf$s_remuser,      netbuf[naf$t_remuser],
3083      3156      2      ***      naf$s_localuser,      netbuf[naf$t_localuser]);
3084      3157      2      6,      netbuf[naf$t_node],
3085      3158      2      12,      netbuf[naf$t_remuser],
3086      3159      2      12,      netbuf[naf$t_localuser]);
3087      3160      2
3088      3161      1      end;
```

```
20 65 74 6F 6D 65 52 20 20 20 65 64 6F 4E 22 00254 P.ACN: .BYTE 34
20 6C 61 63 6F 4C 20 20 20 20 20 72 65 73 55 00255 .ASCII \ Node Remote User Local User\
20 6C 61 63 6F 4C 20 20 20 20 20 72 65 73 55 00264
20 20 20 20 44 41 32 31 21 3A 3A 44 41 36 14 00277 P.ACO: .BYTE 20
20 20 20 20 44 41 32 31 21 3A 3A 44 41 36 21 00278 .ASCII \!6AD::!12AD !12AD\
20 20 20 20 44 41 32 31 21 3A 3A 44 41 36 21 00287
```

```
NAFHDR=      P.ACN
SHOWNAF=      P.ACO
      .EXTRN SYSS$FAO
```

```
      .PSECT $CODE$,NOWRT,2
```

```
OFFC 00000 DISPLAY_PROXY:
```

```
5B 0C000000' 00 9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      3095
5A 00000000G 00 9E 00009      MOVAB      HEADER_FLAG, R11
59 00000000' 00 9E 00010      MOVAB      SYSS$PUT, R10
58 00000000' 00 9E 00017      MOVAB      DBL COLON+4, R9
5E      BC AE 9E 0001E      MOVAB      RABPTR, R8
47      60 AB E9 00022      MOVAB      -68(SP), SP
060C C8 20 20 3A 00026      BLBC      WILD_NETUSER, 1$      3123
56 20 20 50 C3 0002C      LOCC      #32, #32, NETBUF      3131
062C C8 20 20 3A 00030      SUBL3     R0, #32, NODELEN
57 20 20 50 C3 00036      LOCC      #32, #32, NETBUF+32      3132
6E 060C C8 20 20 50 C3 0003A      SUBL3     R0, #32, USRLEN
50 56 28 0003A      MOVAB     NODELEN, NETBUF, PROXY_BUF      3133
6E46 69 D0 00040      MOVAB     DBL COLON+4, R0      3134
9E 60 B0 00043      PUSHAB   PROXY_BUF[NODELEN]
02 AE46 062C C8 57 28 00046      MOVW      (R0), -2(SP)+
55 18 A8 9E 00049      MOVAB     USRLEN, NETBUF+32, PROXY_BUF+2[NODELEN]      3135
53 6E 9E 00051      MOVAB     2(USRLEN)[NODELEN], USRLEN      3137
54 5C A8 9E 00056      MOVAB     MATCH_TOKEN, R5      3138
52 57 D0 0005A      MOVAB     PROXY_BUF, R3
0748 6B 50 E9 0005D      MOVL      MATCH_TOKENLEN, R4
2F 01 D0 00061      MOVL      USRLEN, R2
F8 AB 0140 C9 9B 00064      JSB       FMG$MATCH_NAME
0748 C8 01 D0 0006A      BLBC      R0, 3$      3143
2F 6B E9 00072      MOVL      #1, FOUND_MATCH      3145
F8 AB 0140 C9 9B 00075      BLBC      HEADER_FLAG, 2$      3148
F8 AB 0140 C9 9B 00075      MOVZBW    NAFHDR, FAODSC
```



	FC	A8	0141	C9	9E	0007B	MOVAB	NAFHDR+1, FAODSC+4	
			F0	A8	9F	00081	PUSHAB	DISDSC	
7E		68		22	C1	00084	ADDL3	#34, RABPTR, -(SP)	
			F8	A8	9F	00088	PUSHAB	FAODSC	
00000000G		00		03	FB	0008B	CALLS	#3, SYSS\$FAO	
		6A		68	DD	00092	PUSHL	RABPTR	
		50		01	FB	00094	CALLS	#1, SYSS\$PUT	
			22	68	DD	00097	MOVL	RABPTR, R0	
				A0	B4	0009A	CLRW	34(R0)	
		6A		50	DD	0009D	PUSHL	R0	
				01	FB	0009F	CALLS	#1, SYSS\$PUT	
F8	A8		0163	6B	D4	000A2	CLRL	HEADER FLAG	3150
FC	A8		0164	C9	9B	000A4	MOVZBW	SHOWNAF, FAODSC	3159
			064C	C9	9E	000AA	MOVAB	SHOWNAF+1, FAODSC+4	
				C8	9F	000B0	PUSHAB	NETBUF+64	
			062C	0C	DD	000B4	PUSHL	#12	
				C8	9F	000B6	PUSHAB	NETBUF+32	
			060C	0C	DD	000BA	PUSHL	#12	
				C8	9F	000BC	PUSHAB	NETBUF	
				06	DD	000C0	PUSHL	#6	
			F0	A8	9F	000C2	PUSHAB	DISDSC	
7E		68		22	C1	000C5	ADDL3	#34, RABPTR, -(SP)	
			F8	A8	9F	000C9	PUSHAB	FAODSC	
00000000G		00		09	FB	000CC	CALLS	#9, SYSS\$FAO	
		6A		68	DD	000D3	PUSHL	RABPTR	
				01	FB	000D5	CALLS	#1, SYSS\$PUT	
				04	000D8	3s:	RET		3161

; Routine Size: 217 bytes, Routine Base: \$CODE\$ + 133D

wild\_user - user wild card routine

```
3090 1 %sbttl 'wild_user - user wild card routine'
3091 1 global routine wild_user (action) =
3092 1 begin
3093 1
3094 1 ++
3095 1
3096 1 FUNCTIONAL DESCRIPTION:
3097 1
3098 1 Provide a general means of accessing the User Authorization File
3099 1 records. There are six methods:
3100 1
3101 1 UGMS
3102 1 IRET
3103 1 CPMR
3104 1 \\\
3105 1 FWWW
3106 1 LIII
3107 1 ALLL
3108 1 GDDD
3109 1 Syntax Interpretation
3110 1 -----
3111 1 Username FFFF Exactly one user is to be located
3112 1 * FFFT All users (alphabetically)
3113 1 [Group,Member] TFFF All users with the specified UIC
3114 1 [Group,*] TFTF All users in the specified group (by member)
3115 1 [*,Member] TTFF A FIFO listing of the groups with this member
3116 1 [*,*] TTTF All users by UIC
3117 1
3118 1 INPUTS:
3119 1
3120 1 ACTION - Pointer to routine to call after each successful GET
3121 1
3122 1 IMPLICIT INPUTS:
3123 1
3124 1 UIC_FLAG - UIC form (instead of username)
3125 1 GRP_WILD - Group wild card (must imply UIC_FLAG)
3126 1 MEM_WILD - Member wild card (must imply UIC_FLAG)
3127 1 STR_WILD - all users alphabetically (must imply NOT UIC_FLAG)
3128 1 UAFRAB - RMS data structure for SYSUAF.DAT
3129 1 RECBUF - The current record
3130 1
3131 1 OUTPUTS:
3132 1
3133 1 none
3134 1
3135 1 IMPLICIT OUTPUTS:
3136 1
3137 1 none
3138 1
3139 1 ROUTINE VALUE:
3140 1
3141 1 If an abnormal condition is encountered the appropriate status
3142 1 is returned.
3143 1
3144 1 SIDE EFFECTS:
3145 1
3146 1 none
3147 1 --
```

```
3147 3219 2
3148 3220
3149 3221 macro
3150 3222     lmt_l_uic = 0,0,32,0%;      ! User ID Code
3151 3223     lmt_w_mem = 0,0,16,0%;     ! Member subfield
3152 3224     lmt_w_grp = 2,0,16,0%;     ! Group subfield
3153 3225
3154 3226 local
3155 3227     status,                    ! This routine's status
3156 3228     lmtkey : block[4,byte];    ! Limiting key value for sequential loop
3157 3229
3158 3230 if .uic_flag
3159 3231 then
3160 3232     Change the key of reference and the key buffer if a UIC form was
3161 3233     specified.
3162 3234
3163 3235     begin
3164 3236         uafrab[rab$b_krf] = 1;
3165 3237         uafrab[rab$l_kbf] = recbuf[uaf$l_uic];
3166 3238         uafrab[rab$b_ksz] = 4;
3167 3239     end;
3168 3240
3169 3241 if .mem_wild and not .grp_wild
3170 3242 then
3171 3243     The UIC requested was of the form [Group,*]
3172 3244
3173 3245     uafrab[rab$v_kge] = true;
3174 3246
3175 3247
3176 3248
3177 3249     LMTKEY need be loaded only IF .UIC FLAG AND NOT (.GRP_WILD AND .MEM_WILD)
3178 3250     but it is simpler to always load it.
3179 3251
3180 3252
3181 3253     lmtkey[lmt_l_uic] = .recbuf[uaf$l_uic];
3182 3254
3183 3255
3184 3256     Locate the first user meeting the specification.
3185 3257
3186 3258
3187 3259 if .str_wild or .grp_wild
3188 3260 then
3189 3261     begin
3190 3262
3191 3263         Every user in the file is to be accessed.
3192 3264
3193 3265         uafrab[rab$b_rac] = rab$c_seq;
3194 3266         $rewind (rab = uafrab);
3195 3267         status = get_uaf_record ();
3196 3268     end
3197 3269 else
3198 3270     begin
3199 3271         status = get_uaf_record ();
3200 3272         if .uic_flag
3201 3273         then
3202 3274             begin
3203 3275
```

wild\_user - user wild card routine

```
3204 3276 4 | Even an explicit UIC requires sequential reads to locate duplicates.
3205 3277 4 |
3206 3278 4 |     uafrab[ra$b_rac] = rab$c_seq;
3207 3279 4 |     if .mem_wild and not .grp_wild
3208 3280 4 |     then
3209 3281 4 |         begin
3210 3282 4 |
3211 3283 4 |         RABSV_KGE is set on the initial access for specifications of the
3212 3284 4 |         form [Group,*] so if the specified group has no members the record
3213 3285 4 |         will be that of a user in another group.
3214 3286 4 |
3215 3287 4 |         uafrab[ra$b_kge] = false;
3216 3288 4 |         if .lmtkey[lmt_w_grp] nequ .recbuf[uafrab$w_grp]
3217 3289 4 |         then
3218 3290 4 |             status = rms$rnf;
3219 3291 4 |         end;
3220 3292 4 |     end;
3221 3293 4 | end;
3222 3294 4 |
3223 3295 4 | if .status
3224 3296 4 | then
3225 3297 4 |     begin
3226 3298 4 |
3227 3299 4 |     Feed the action routine the first record. In the case of an explicit
3228 3300 4 |     username specification this will be the only record.
3229 3301 4 |
3230 3302 4 |     while .status
3231 3303 4 |     do
3232 3304 4 |         begin
3233 3305 4 |             if (
3234 3306 4 |                 if .grp_wild and not .mem_wild
3235 3307 4 |                 then .recbuf[uafrab$w_mem] eql .lmtkey[lmt_w_mem]
3236 3308 4 |                 else true
3237 3309 4 |             )
3238 3310 4 |             then status = (.action) ();
3239 3311 4 |             if not .status then exitloop;
3240 3312 4 |             if not (.str_wild or .uic_flag)
3241 3313 4 |             then exitloop;
3242 3314 4 |             status = get uaf record ();
3243 3315 4 |             if not .status then exitloop;
3244 3316 4 |             if .uic_flag
3245 3317 4 |             then
3246 3318 4 |                 begin
3247 3319 4 |
3248 3320 4 |                 The limiting key value is used in different ways depending
3249 3321 4 |                 on the form of the UIC specification.
3250 3322 4 |
3251 3323 4 |                 if .mem_wild and not .grp_wild
3252 3324 4 |                 then
3253 3325 4 |
3254 3326 4 |                 [Group,*]
3255 3327 4 |
3256 3328 4 |                 begin
3257 3329 4 |                     if .lmtkey[lmt_w_grp] nequ .recbuf[uafrab$w_grp]
3258 3330 4 |                     then exitloop;
3259 3331 4 |                 end;
3260 3332 4 |                 if not (.grp_wild or .mem_wild)
```

wild\_user - user wild card routine

```
3261      then
3262      [Group,Member]
3263      begin
3264      if .lmtkey[lmt_l_uic] nequ .recbuf[uaf$l_uic]
3265      then exitloop;
3266      end;
3267      end;
3268      if .status eql rms$_eof
3269      then status = true;
3270      end;
3271      ! end of wild carding loop
3272      ! Hitting EOF is ok
3273      if .str_wild and not .found_match
3274      then LIB$SIGNAL(UAF$_BADSPCT);
3275      The RAB must be returned to its former state before exiting.
3276      uafrab[rab$b_rac] = rab$c_key;
3277      if .uic_flag
3278      then
3279      begin
3280      uafrab[rab$b_krf] = 0;
3281      uafrab[rab$l_kbf] = recbuf[uaf$t_username];
3282      uafrab[rab$b_ksz] = uaf$s_username;
3283      end;
3284      Reset parse count
3285      call_count = 0;
3286      .status
3287      end;
```

	54	00000000V	00	001C	00000	.ENTRY	WILD USER, Save R2,R3,R4	3163
	53	00000000'	00	9E	00002	MOVAB	GET_OAF_RECORD, R4	
	5E		04	C2	00010	MOVAB	GRP_WILD, R3	
	0E	FC	A3	E9	00013	SUBL2	#4, SP	
FF50	C3	F95C	C3	9E	00017	BLBC	UIC_FLAG, 1\$	3229
FF54	C3	0104	8F	B0	0001E	MOVAB	RECBUF+36, UAFRAB+48	3237
	08	04	A3	E9	00025	MOVW	#260, UAFRAB+52	3238
	05		63	E8	00029	BLBC	MEM_WILD, 2\$	3241
FF26	C3		20	88	0002C	BLBS	GRP_WILD, 2\$	
	6E	F95C	C3	D0	00031	BISB2	#32, UAFRAB+6	3246
	03	08	A3	E8	00036	MOVL	RECBUF+36, LMTKEY	3253
	17		63	E9	0003A	BLBS	STR_WILD, 3\$	3259
		FF3E	C3	94	0003D	BLBC	GRP_WILD, 4\$	
		FF20	C3	9F	00041	CLRB	UAFRAB+30	3265
						PUSHAB	UAFRAB	3266



00000000G	00	01	FB	00045	CALLS	#1, SYSSREWIND	..	
	64	00	FB	0004C	CALLS	#0, GET_UAF_RECORD	..	3267
	52	50	D0	0004F	MOVL	R0, STATUS	..	
		29	11	00052	BRB	5%	..	3259
	64	00	FB	00054	CALLS	#0, GET_UAF_RECORD	..	3271
	52	50	D0	00057	MOVL	R0, STATUS	..	
	1F	A3	E9	0005A	BLBC	UIC_FLAG, 5%	..	3272
		C3	94	0005E	CLRB	UAFRAB+30	..	3278
	17	A3	E9	00062	BLBC	MEM_WILD, 5%	..	3279
	14	63	E8	00066	BLBS	GRP_WILD, 5%	..	
FF26	C3	20	8A	00069	BICB2	#32, UAFRAB+6	..	3287
F95E	C3	AE	B1	0006E	CMPW	LMTKEY+2, RECBUF+38	..	3288
		07	13	00074	BEQL	5%	..	
	52	8F	D0	00076	MOVL	#98994, STATUS	..	3290
	58	52	E9	0007D	BLBC	STATUS, 12%	..	3295
	4C	52	E9	00080	BLBC	STATUS, 11%	..	3302
	0B	63	E9	00083	BLBC	GRP_WILD, 7%	..	3306
	07	A3	E8	00086	BLBS	MEM_WILD, 7%	..	
	6E	C3	B1	0008A	CMPW	RECBUF+36, LMTKEY	..	3307
		07	12	0008F	BNEQ	8%	..	
04	BC	00	FB	00091	CALLS	#0, @ACTION	..	3310
	52	50	D0	00095	MOVL	R0, STATUS	..	
	34	52	E9	00098	BLBC	STATUS, 11%	..	3311
	04	A3	E8	0009B	BLBS	STR_WILD, 9%	..	3312
	2C	A3	E9	0009F	BLBC	UIC_FLAG, 11%	..	
	64	00	FB	000A3	CALLS	#0, GET_UAF_RECORD	..	3314
	52	50	D0	000A6	MOVL	R0, STATUS	..	
	23	52	E9	000A9	BLBC	STATUS, 11%	..	3315
	D0	A3	E9	000AC	BLBC	UIC_FLAG, 6%	..	3316
	50	A3	D0	000B0	MOVL	MEM_WILD, R0	..	3323
	0B	50	E9	000B4	BLBC	R0, 10%	..	
	C6	63	E8	000B7	BLBS	GRP_WILD, 6%	..	
F95E	C3	AE	B1	000BA	CMPW	LMTKEY+2, RECBUF+38	..	3329
		0D	12	000C0	BNEQ	11%	..	
	BB	63	E8	000C2	BLBS	GRP_WILD, 6%	..	3332
	B8	50	E8	000C5	BLBS	R0, 6%	..	
F95C	C3	6E	D1	000C8	CMPW	LMTKEY, RECBUF+36	..	3338
		B1	13	000CD	BEQL	6%	..	
0001827A	8F	52	D1	000CF	CMPW	STATUS, #98938	..	3343
		03	12	000D6	BNEQ	12%	..	
	52	01	D0	000D8	MOVL	#1, STATUS	..	3344
	11	A3	E9	000DB	BLBC	STR_WILD, 13%	..	3347
	0D	A3	E8	000DF	BLBS	FOUND_MATCH, 13%	..	
		8F	DD	000E3	PUSHL	#UAF\$-BADSPC	..	3348
00000000G	00	01	FB	000E9	CALLS	#1, LTB\$SIGNAL	..	
FF3E	C3	01	90	000F0	MOVB	#1, UAFRAB+30	..	3353
	0C	A3	E9	000F5	BLBC	UIC_FLAG, 14%	..	3355
FF50	C3	C3	9E	000F9	MOVAB	RECBUF+4, UAFRAB+48	..	3359
FF54	C3	20	B0	00100	MOVW	#32, UAFRAB+52	..	3360
		C3	D4	00105	CLRL	CALL COUNT	..	3366
	50	52	D0	00109	MOVL	STATUS, R0	..	3369
		04	0010C	RET			..	

; Routine Size: 269 bytes, Routine Base: \$CODE\$ + 1416

```
3299 3370 1 %sbttl 'display_brief - writes a brief user display'
3300 3371 routine display_brief =
3301 3372 begin
3302 3373
3303 3374 ++
3304 3375
3305 3376 FUNCTIONAL DESCRIPTION:
3306 3377
3307 3378     Provide an ASCII listing of the most important record information
3308 3379     (username, owner, etc.) for each record supplied.
3309 3380
3310 3381 INPUTS:
3311 3382
3312 3383     none
3313 3384
3314 3385 IMPLICIT INPUTS:
3315 3386
3316 3387     RABPTR - RMS data structure for the file
3317 3388
3318 3389 OUTPUTS:
3319 3390
3320 3391     none
3321 3392
3322 3393 IMPLICIT OUTPUTS:
3323 3394
3324 3395     none
3325 3396
3326 3397 ROUTINE VALUE:
3327 3398
3328 3399     none
3329 3400
3330 3401 SIDE EFFECTS:
3331 3402
3332 3403     none
3333 3404
3334 3405 --
3335 3406
3336 3407 P bind
3337 3408     lststr1 = cstring ('      Owner      Username      UIC      Account Privs',
3338 3409     'Pri Directory'),
3339 3410     lststr2 = cstring ('!20AC !12AD !15%U !8AF !6AC !2UL !AC!AC');
3340 3411
3341 3412     Output a header if one was requested.
3342 3413
3343 3414     if .header_flag
3344 3415     then
3345 3416     begin
3346 3417         faomac (lststr1);
3347 3418         output_null;
3348 3419         header_flag = false;
3349 3420     end;
3350 3421
3351 3422     if .str_wild and not fmg$match_name (namelen (uaf$s_username, recbuf[uaf$t_username]),
3352 3423     recbuf[uaf$t_username],
3353 3424     .match_token(en, match_token)
3354 3425
3355 3426     then return true;
```

```

3356 found_match = true;
3357
3358 |
3359 | Output the record.
3360 |
3361 |
3362 |
3363 |
3364 |
3365 |
3366 |
3367 |
3368 |
3369 |
3370 |
3371 |
3372 |
3373 |
3374 |
3375 |
3376 |
3377 |

```

[illegible]

				.PSECT		SCODES, NOWRT, 2	
				07FC 00000 DISPLAY_BRIEF:			
				.WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10			
				MOVAB HEADER FLAG, R10			
				MOVAB SYSS\$AD, R9			
				MOVAB SYSS\$PUT, R8			
				MOVAB LSTSTR1, R7			
				MOVAB RABPTR, R6			
				BLBC HEADER FLAG, 18			
				MOVZBW LSTSTR1, FAODSC			
				MOVAB LSTSTR1+1, FAODSC+4			
				PUSHAB DISDSC			
				ADDL3 #34, RABPTR, -(SP)			
				3371			
				3414			
				3417			

; Routine Size: 217 bytes, Routine Base: \$CODE\$ + 1523



```
3379 1 %sbttl 'classify_priv - classifies contents of priv vector'
3380 1 routine classify_priv (privadr, uic) =
3381 2 begin
3382 2 ++
3383 2
3384 2
3385 2 FUNCTIONAL DESCRIPTION:
3386 2
3387 2     Classifies privilege bits and reports the highest class available
3388 2     to the owner of the supplied vector.
3389 2
3390 2 INPUTS:
3391 2
3392 2     PRIVADR - Address of the privilege vector
3393 2
3394 2 IMPLICIT INPUTS:
3395 2
3396 2     none
3397 2
3398 2 OUTPUTS:
3399 2
3400 2     none
3401 2
3402 2 IMPLICIT OUTPUTS:
3403 2
3404 2     none
3405 2
3406 2 ROUTINE VALUE:
3407 2
3408 2     none
3409 2
3410 2 SIDE EFFECTS:
3411 2
3412 2     none
3413 2
3414 2 --
3415 2 map
3416 2     privadr : ref block[8,byte];
3417 2
3418 2 bind
3419 2     lstprva = cstring ('All'),
3420 2     lstprvb = cstring ('Files'),
3421 2     lstprvc = cstring ('System'),
3422 2     lstprvd = cstring ('Devour'),
3423 2     lstprve = cstring ('Group'),
3424 2     lstprvf = cstring ('Normal'),
3425 2     lstprvg = cstring ('None');
3426 2
3427 2 if .privadr[priv$u_cmkrnl]
3428 2 or .privadr[priv$u_cmexec]
3429 2 or .privadr[priv$u_sysnam]
3430 2 or .privadr[priv$u_detach]
3431 2 or .privadr[priv$u_log_io]
3432 2 or .privadr[priv$u_setprv]
3433 2 or .privadr[priv$u_phy_io]
3434 2 or .privadr[priv$u_pfnmap]
3435 2 or .privadr[priv$u_sysprv]
```



classify\_priv - classifies contents of priv vec

```
3436 3506 2 or .privadr[priv$readall]
3437 3507 or .privadr[priv$bypass]
3438 3508 or (.uic<16, 16> lequ .EXESGL_SYSUIC)
3439 3509 then return lstprva; ! Universal Privilege
3440 3510
3441 3511 if .privadr[priv$diagnose]
3442 3512 or .privadr[priv$volpro]
3443 3513 or .privadr[priv$upgrade]
3444 3514 or .privadr[priv$downgrade]
3445 3515 or .privadr[priv$security]
3446 3516 or .privadr[priv$sysgbl]
3447 3517 then return lstprvb; ! Potentially Compromises File Security
3448 3518
3449 3519 if .privadr[priv$pswapm]
3450 3520 or .privadr[priv$setpri]
3451 3521 or .privadr[priv$world]
3452 3522 or .privadr[priv$oper]
3453 3523 then return lstprvc; ! Can Interfere with System Operation
3454 3524
3455 3525 if .privadr[priv$grpnam]
3456 3526 or .privadr[priv$allspool]
3457 3527 or .privadr[priv$noact]
3458 3528 or .privadr[priv$prmceb]
3459 3529 or .privadr[priv$prmbbx]
3460 3530 or .privadr[priv$exquota]
3461 3531 or .privadr[priv$bugchk]
3462 3532 or .privadr[priv$prmgbl]
3463 3533 or .privadr[priv$shmem]
3464 3534 then return lstprvd; ! Can Devour System Resources
3465 3535
3466 3536 if .privadr[priv$group]
3467 3537 or .privadr[priv$grppriv]
3468 3538 then return lstprve; ! Can Interfere with Group Members
3469 3539
3470 3540 if .privadr[priv$tmpmbx]
3471 3541 or .privadr[priv$netmbx]
3472 3542 or .privadr[priv$mount]
3473 3543 then return lstprvf; ! Normal Privileges
3474 3544
3475 3545 lstprvg ! Not Privileged
3476 3546 end;
```

.PSECT SPLITS, NOWRT, NOEXE, 2

				03	00303	P.ACR:	.BYTE	3		
		6C	6C	41	00304		.ASCII	\ALL\		
				05	00307	P.ACS:	.BYTE	5		
		73	65	6C	69	46	00308	.ASCII	\Files\	
				06	0030D	P.ACT:	.BYTE	6		
		6D	65	74	73	79	53	0030E	.ASCII	\System\
				06	00314	P.ACU:	.BYTE	6		
		72	75	6F	76	65	44	00315	.ASCII	\Devour\
				05	0031B	P.ACV:	.BYTE	5		
		70	75	6F	72	47	0031C	.ASCII	\Group\	
				06	00321	P.ACW:	.BYTE	6		

6C 61 6D 72 6F 4E 00322 .ASCII \Normal\  
04 00328 P.ACX: .BYTE 4  
65 6E 6F 4E 00329 .ASCII \None\

LSTPRVA=  
LSTPRVB=  
LSTPRVC=  
LSTPRVD=  
LSTPRVE=  
LSTPRVF=  
LSTPRVG=  
P.ACR  
P.ACS  
P.ACT  
P.ACU  
P.ACV  
P.ACW  
P.ACX

.PSECT \$CODE\$,NOWRT,2

				0004 00000 CLASSIFY PRIV:				
		52	00000000'	00	9E	00002	Save R2	3450
		50	04	AC	DO	00009	MOVAB LSTPRVA, R2	3497
		35		60	E8	00000	MOVL PRVADR, R0	
31		60		01	E0	00010	BLBS (R0), 1\$	3498
2D		60		02	E0	00014	BBS #1, (R0), 1\$	3499
29		60		05	E0	00018	BBS #2, (R0), 1\$	3500
				60	95	0001C	BBS #5, (R0), 1\$	3501
				25	19	0001E	TSTB (R0)	
21		60		0E	E0	00020	BLSS 1\$	3502
1D		60		16	E0	00024	BBS #14, (R0), 1\$	3503
19		60		1A	E0	00028	BBS #22, (R0), 1\$	3504
15		60		1C	E0	0002C	BBS #26, (R0), 1\$	3505
10	04	60		03	E0	00030	BBS #28, (R0), 1\$	3506
0C		60		1D	E0	00035	BBS #3, 4(R0), 1\$	3507
00000000G 00	0A	10		00	ED	00039	BBS #29, (R0), 1\$	3508
				05	1A	00043	CMPZV #0, #16, UIC+2, EXE\$GL_SYSUIC	
		51		62	9E	00045	BGTRU 2\$	3509
				6D	11	00048	MOVAB LSTPRVA, R1	
16		60		06	E0	0004A	BRB 10\$	3511
12		60		15	E0	0004E	BBS #6, (R0), 3\$	3512
		0E	04	A0	E8	00052	BBS #21, (R0), 3\$	3513
09	04	A0		01	E0	00056	BLBS 4(R0), 3\$	3514
04	04	A0		06	E0	0005B	BBS #1, 4(R0), 3\$	3515
06		60		19	E1	00060	BBS #6, 4(R0), 3\$	3516
		51	04	A2	9E	00064	BBC #25, (R0), 4\$	3517
				4D	11	00068	MOVAB LSTPRVB, R1	
0C		60		0C	E0	0006A	BRB 10\$	3519
08		60		0D	E0	0006E	BBS #12, (R0), 5\$	3520
		04	02	A0	E8	00072	BBS #13, (R0), 5\$	3521
06		60		12	E1	00076	BLBS 2(R0), 5\$	3522
		51	0A	A2	9E	0007A	BBC #18, (R0), 6\$	3523
				37	11	0007E	MOVAB LSTPRVC, R1	
20		60		03	E0	00080	BRB 10\$	3525
1C		60		04	E0	00084	BBS #3, (R0), 7\$	3526
18		60		09	E0	00088	BBS #4, (R0), 7\$	3527
14		60		0A	E0	0008C	BBS #9, (R0), 7\$	3528
10		60		0B	E0	00090	BBS #10, (R0), 7\$	3529
0C		60		13	E0	00094	BBS #11, (R0), 7\$	3530
08		60		17	E0	00098	BBS #19, (R0), 7\$	3531
		04	03	A0	E8	0009C	BBS #23, (R0), 7\$	3532
							BLBS 3(R0), 7\$	

UAFMAIN  
V04-000

classifi\_priv - classifies contents of priv vec

B 4  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 124  
(26)

06	60	11	1B	E1	000A0	BBC	#27, (R0), 8\$	3533
	51		A2	9E	000A4	MOVAB	LSTPRVD, R1	3534
			0D	11	000A8	BRB	10\$	
	05	01	A0	E8	000AA	BLBS	1(R0), 9\$	3536
08	A0		02	E1	000AE	BBC	#2, 4(R0), 11\$	3537
	51	18	A2	9E	000B3	MOVAB	LSTPRVE, R1	3538
	50		51	D0	000B7	MOVL	R1, R0	
				04	000BA	RET		
			60	B5	000BB	TSTW	(R0)	3540
			08	19	000BD	BLSS	12\$	
04	60		14	E0	000BF	BBS	#20, (R0), 12\$	3541
05	60		11	E1	000C3	BBC	#17, (R0), 13\$	3542
	50	1E	A2	9E	000C7	MOVAB	LSTPRVF, R0	3543
				04	000CB	RET		
	50	25	A2	9E	000CC	MOVAB	LSTPRVG, R0	3451
			04	000D0	RET			3546

; Routine Size: 209 bytes, Routine Base: \$CODE\$ + 15FC

UAF  
V04

display\_full - writes the full user display

```
1 %sbttl 'display_full - writes the full user display'
1 routine display_full =
2 begin
```

++

## FUNCTIONAL DESCRIPTION:

Display the fields of a UAF record.

## INPUTS:

RABPTR - RMS data structure for the file

## IMPLICIT INPUTS:

none

## OUTPUTS:

none

## IMPLICIT OUTPUTS:

none

## ROUTINE VALUE:

none

## SIDE EFFECTS:

none

--

Display strings.

## local

status : long ;

## bind

```
username = cstring ('Username: !32AF Owner: !AC'),
account = cstring ('Account: !32AF UIC: !XU (!XI)'),
cli_table = cstring ('CLI: !32AC Tables: !AC'),
default = cstring ('Default: !AC!AC'),
lgicmd = cstring ('LGICMD: !AC'),
flags = cstring ('Login Flags: !AD'),
flag_pad = cstring ('%char (cr), %char ((f), '),
primdays = cstring ('Primary days: !7(AC)'),
secdays = cstring ('Secondary days: !7(AC)'),
norestrict = cstring ('No access restrictions'),
accesshdr1 = cstring ('
Primary 000000000011111111112222 Secondary 000000000011111111112222'),
accesshdr2 = cstring ('
Day Hours 012345678901234567890123 Day Hours 012345678901234567890123'),
```



```
3535 3604 2 netaccess = cstring ('Network: !AD !AD'),
3536 3605 bataccess = cstring ('Batch: !AD !AD'),
3537 3606 locaccess = cstring ('Local: !AD !AD'),
3538 3607 diaaccess = cstring ('Dialup: !AD !AD'),
3539 3608 remaccess = cstring ('Remote: !AD !AD'),
3540 3609 expiration = cstring ('Expiration: !AD Pwdminimum: !2UL Login Fails: !5UL'),
3541 3610 pwddata = cstring ('Pwdlifetime: !AD Pwdchange: !AD !AD'),
3542 3611 lastlogin = cstring ('Last Login: !AD (interactive), !AD (non-interactive)'),
3543 3612 quota1 = cstring ('Maxjobs: !5UL Fillm: !5UL Bytlm: !9UL'),
3544 3613 quota2 = cstring ('Maxacctjobs: !5UL Shrfillm: !5UL Pbytlm: !9UL'),
3545 3614 quota3 = cstring ('Maxdetach: !5UL B10lm: !5UL JTquota: !9UL'),
3546 3615 quota4 = cstring ('Prclm: !5UL D10lm: !5UL WSdef: !9UL'),
3547 3616 quota5 = cstring ('Prio: !5UL ASTlm: !5UL WSquo: !9UL'),
3548 3617 quota6 = cstring ('Queprio: !5UL TQElm: !5UL WSextent: !9UL'),
3549 3618 quota7 = cstring ('CPU: !13AD Enqlm: !5UL Pqflquo: !9UL'),
3550 3619 privs = cstring ('Authorized Privileges: '),
3551 3620 defprivs = cstring ('Default Privileges: '),
3552 3621 nullstr = cstring (''),
3553 3622
3554 3623
3555 3624 mon = cstring (' Mon'),
3556 3625 tue = cstring (' Tue'),
3557 3626 wed = cstring (' Wed'),
3558 3627 thu = cstring (' Thu'),
3559 3628 fri = cstring (' Fri'),
3560 3629 sat = cstring (' Sat'),
3561 3630 sun = cstring (' Sun'),
3562 3631 noday = cstring (' '),
3563 3632
3564 3633 cputime = recbuf[uaf$l_cputim]; ! CPU Limit in hundredths of a second
3565 3634
3566 3635 own
3567 3636 flags_vector : vector [32] preset (
3568 3637 [$bitposition (uaf$V_audit)] = cstring (' Audit'),
3569 3638 [$bitposition (uaf$V_captive)] = cstring (' Captive'),
3570 3639 [$bitposition (uaf$V_defcli)] = cstring (' Defcli'),
3571 3640 [$bitposition (uaf$V_disctly)] = cstring (' Disctly'),
3572 3641 [$bitposition (uaf$V_nomail)] = cstring (' Dismail'),
3573 3642 [$bitposition (uaf$V_dismail)] = cstring (' Disnewmail'),
3574 3643 [$bitposition (uaf$V_disreconnect)] = cstring (' Disreconnect'),
3575 3644 [$bitposition (uaf$V_disreport)] = cstring (' Disreport'),
3576 3645 [$bitposition (uaf$V_disacct)] = cstring (' Disuser'),
3577 3646 [$bitposition (uaf$V_diswelcom)] = cstring (' Diswelcome'),
3578 3647 [$bitposition (uaf$V_genpwd)] = cstring (' Genpwd'),
3579 3648 [$bitposition (uaf$V_lockpwd)] = cstring (' Lockpwd'),
3580 3649 [$bitposition (uaf$V_pwd_expired)] = cstring (' Pwd_expired'),
3581 3650 [$bitposition (uaf$V_pwd2_expired)] = cstring (' Pwd2_expired'),
3582 3651 );
3583 3652
3584 3653 local
3585 3654 count, ! count for string being built
3586 3655 lcount, ! count of chars on current line
3587 3656 string : vector [160, byte], ! buffer to build display string
3588 3657 flag_string : ref vector [,byte], ! pointer to flag string
3589 3658 delta_time : vector [long, 2], ! Scratch area for system delta time
3590 3659 PTR, ! Pointer into UAF$Q_PWD_DATE quadword
3591 3660 time1 : vector [17, byte], ! buffer for time string
```



display\_full - writes the full user display

```
3592 3661      time2      : vector [17, byte],      ! buffer for time string
3593 3662      time3      : vector [17, byte],      ! buffer for time string
3594 3663
3595 3664 builtin
3596 3665      emul;
3597 3666
3598 3667
3599 3668
3600 3669      if .str_wild and not fmg$match_name (namelen (uaf$s_username, recbuf[uaf$t_username]),
3601 3670                                          recbuf[uaf$t_username],
3602 3671                                          .match_token[en, match_token])
3603 3672      then return true;
3604 3673
3605 3674      found_match = true;
3606 3675
3607 3676      output_null;
3608 3677
3609 3678      faomac (username,
3610 3679            uaf$s_username, recbuf[uaf$t_username],
3611 3680            recbuf[uaf$t_owner]);
3612 3681
3613 3682      faomac (account,
3614 3683            uaf$s_account, recbuf[uaf$t_account],
3615 3684            .recbuf[uaf$l_uic],
3616 3685            .recbuf[uaf$l_uic]);
3617 3686
3618 3687      faomac (cli_table,
3619 3688            recbuf[uaf$t_defcli],
3620 3689            recbuf[uaf$t_clitables]);
3621 3690
3622 3691      faomac (default,
3623 3692            recbuf[uaf$t_defdev],
3624 3693            recbuf[uaf$t_defdir]);
3625 3694
3626 3695      faomac (lgicmd,
3627 3696            recbuf[uaf$t_lgicmd]);
3628 3697
3629 3698      count = 0;
3630 3699      lcount = .flags<0,8>;
3631 3700      incr j from 0 to 31
3632 3701      do
3633 3702          begin
3634 3703              if .bitvector [recbuf[uaf$l_flags], .j]
3635 3704              and (flag_string = .flags_vector [.j]) neq 0
3636 3705              then
3637 3706                  begin
3638 3707                      if .lcount + .flag_string[0] gtru 80
3639 3708                      then
3640 3709                          begin
3641 3710                              ch$move (.flag_pad<0,8>, flag_pad+1, string[.count]);
3642 3711                              count = .count + .flag_pad<0,8>;
3643 3712                              lcount = .flag_pad<0,8> - 2;
3644 3713                              end;
3645 3714                              ch$move (.flag_string[0], flag_string[1], string[.count]);
3646 3715                              count = .count + .flag_string[0];
3647 3716                              lcount = .lcount + .flag_string[0];
3648 3717                          end;
```

```
3649      3718      2      end;
3650      3719      2
3651      P      3720      faomac (flags,
3652      3721      .count, string);
3653      3722
3654      P      3723      faomac (primdays,
3655      P      3724      if .recbuf[uaf$u_monday]      then noday      else mon,
3656      P      3725      if .recbuf[uaf$u_tuesday]      then noday      else tue,
3657      P      3726      if .recbuf[uaf$u_wednesday]      then noday      else wed,
3658      P      3727      if .recbuf[uaf$u_thursday]      then noday      else thu,
3659      P      3728      if .recbuf[uaf$u_friday]      then noday      else fri,
3660      P      3729      if .recbuf[uaf$u_saturday]      then noday      else sat,
3661      3730      if .recbuf[uaf$u_sunday]      then noday      else sun);
3662      3731
3663      P      3732      faomac (secdays,
3664      P      3733      if .recbuf[uaf$u_monday]      then mon      else noday,
3665      P      3734      if .recbuf[uaf$u_tuesday]      then tue      else noday,
3666      P      3735      if .recbuf[uaf$u_wednesday]      then wed      else noday,
3667      P      3736      if .recbuf[uaf$u_thursday]      then thu      else noday,
3668      P      3737      if .recbuf[uaf$u_friday]      then fri      else noday,
3669      P      3738      if .recbuf[uaf$u_saturday]      then sat      else noday,
3670      3739      if .recbuf[uaf$u_sunday]      then sun      else noday);
3671      3740
3672      3741      if ch$fail (ch$find_not_ch (10*uaf$u_network_access_p, recbuf[uaf$b_network_access_p], 0))
3673      3742      then
3674      3743      begin
3675      3744      faomac (norestrict);
3676      3745      end
3677      3746      else
3678      3747      begin
3679      3748      faomac (accesshdr1);
3680      3749      faomac (accesshdr2);
3681      3750
3682      3751      display_hours (netaccess, recbuf[uaf$b_network_access_p]);
3683      3752      display_hours (batchaccess, recbuf[uaf$b_batch_access_p]);
3684      3753      display_hours (locaccess, recbuf[uaf$b_local_access_p]);
3685      3754      display_hours (diaaccess, recbuf[uaf$b_dialup_access_p]);
3686      3755      display_hours (remaccess, recbuf[uaf$b_remote_access_p]);
3687      3756      end;
3688      3757
3689      3758      convert time (recbuf[uaf$q_expiration], 17, time1);
3690      P      3759      faomac Texpiration,
3691      P      3760      17, time1,
3692      P      3761      .recbuf[uaf$b_pwd_length],
3693      3762      .recbuf[uaf$u_logTails]);
3694      3763
3695      3764      convert time (recbuf[uaf$q_pwd_lifetime], 10, time1);
3696      3765      PTR = RECBUF[UAF$Q_PWD_DATE]; ! Because quadwords have 'no' width
3697      3766      if (.PTR eql -1) and T.(PTR+Xupval) eql -1 then
3698      3767      ch$move(17, uplit(%ascii' (pre-expired)'), TIME2)
3699      3768      else
3700      3769      CONVERT_TIME(.PTR 17, TIME2);
3701      3770      convert time (recbuf[uaf$q_pwd2_date], 17, time3);
3702      P      3771      faomac Tpwddata,
3703      P      3772      10, time1,
3704      P      3773      17, time2,
3705      P      3774      (if (. (recbuf[uaf$q_pwd2_date]+0) or . (recbuf[uaf$q_pwd2_date]+4)) eql 0
```

```
3706 P 3775      then 0
3707      3776      else 17), time3);
3708      3777
3709      3778      convert_time (recbuf[uaf$q_lastlogin_i], 17, time1);
3710      3779      convert_time (recbuf[uaf$q_lastlogin_n], 17, time2);
3711      P 3780      faomac (lastlogin,
3712      P 3781      17, time1
3713      3782      17, time2);
3714      3783
3715      3784      emul (%ref (-200000), %ref (.cputime<1 31>)
3716      3785      %ref (if .cputime<0,1> then -100000 else 0), delta_time);
3717      3786      convert_time (delta_time, 13, time1);
3718      3787
3719      P 3788      faomac (quota1,
3720      P 3789      .recbuf[uaf$w_maxjobs],
3721      P 3790      .recbuf[uaf$w_fillm],
3722      3791      .recbuf[uaf$l_byt1m]);
3723      3792
3724      P 3793      faomac (quota2,
3725      P 3794      .recbuf[uaf$w_maxacctjobs],
3726      P 3795      .recbuf[uaf$w_shrfillm],
3727      3796      .recbuf[uaf$l_pbyt1m]);
3728      3797
3729      P 3798      faomac (quota3,
3730      P 3799      .recbuf[uaf$w_maxdetach],
3731      3800      .recbuf[uaf$w_bi1m],
3732      3801      .recbuf[uaf$l_jtquota]);
3733      3802
3734      P 3803      faomac (quota4,
3735      P 3804      .recbuf[uaf$w_prcnt],
3736      P 3805      .recbuf[uaf$w_diolm],
3737      3806      .recbuf[uaf$l_dfwsct]);
3738      3807
3739      P 3808      faomac (quota5,
3740      P 3809      .recbuf[uaf$b_pri],
3741      P 3810      .recbuf[uaf$w_ast1m],
3742      3811      .recbuf[uaf$l_wsquota]);
3743      3812
3744      P 3813      faomac (quota6,
3745      P 3814      .recbuf[uaf$b_quepri],
3746      P 3815      .recbuf[uaf$w_tqcnt],
3747      3816      .recbuf[uaf$l_wsextent]);
3748      3817
3749      P 3818      faomac (quota7,
3750      P 3819      13, time1,
3751      P 3820      .recbuf[uaf$w_eng1m],
3752      3821      .recbuf[uaf$l_pg1quota]);
3753      3822
3754      3823      faomac (privs);
3755      3824      print_priv (recbuf[uaf$q_priv]);
3756      3825
3757      3826      faomac (defprivs);
3758      3827      print_priv (recbuf[uaf$q_def_priv]);
3759      3828
3760      3829      !
3761      3830      ! Build a holder from the UAF record and display the rights
3762      3831      ! granted to it.
```

display\_full - writes the full user display

H 4  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 130  
(27)

```

3763 3832 2 1
3764 3833 2 1
3765 3834 2 1
3766 3835 2 1
3767 3836 2 1
3768 3837 2 1
3769 3838 2 1
3770 3839 2 1
3771 3840 2 1
3772 3841 2 1
3773 3842 2 1

```

```

1
if .rdb_exists
then
begin
uaf$build_holder ();
rdb_header_flag = true ;
status = uaf$write_rights ( holder ) ;
end ;
return .status ;
end;

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
46 41 32 33 21 20 3A 65 6D 61 6E 72 65 73 1B 0032D P.ACY: .BYTE 27
43 41 21 20 20 3A 72 65 6E 77 4F 20 0032E .ASCII \Username: !32AF Owner: !AC\
21 28 20 55 25 21 20 20 3A 74 6E 75 6F 63 63 21 0033D P.ACZ: .BYTE 33
21 28 20 55 25 21 20 20 3A 74 6E 75 6F 63 63 41 00349 .ASCII \Account: !32AF UIC: !%U (!%I)\
29 49 25 00359
18 00368 P.ADA: .BYTE 27
43 41 21 20 20 3A 73 65 6C 62 61 54 43 0036C .ASCII \CLI: !32AC Tables: !AC\
20 0037B
41 21 43 41 21 20 20 3A 74 6C 75 61 66 65 44 00387 P.ADB: .BYTE 16
43 00388 .ASCII \Default: !AC!AC\
0D 00397
43 41 21 20 20 20 3A 44 4D 43 49 47 4C 00398 P.ADC: .BYTE 13
10 00399 .ASCII \LGICMD: !AC\
41 21 20 3A 73 67 61 6C 46 20 6E 69 67 6F 4C 003A6 P.ADD: .BYTE 16
44 003A7 .ASCII \Login Flags: !AD\
0F 003B6
20 20 20 20 20 20 20 20 20 20 20 20 20 0A 0D 003B7 P.ADE: .BYTE 15
15 003B8 .ASCII <13><10>\
20 20 3A 73 79 61 64 20 79 72 61 6D 69 72 50 003C7 P.ADF: .BYTE 21
21 003C8 .ASCII \Primary days: !7(AC)\
15 003D7
3A 73 79 61 64 20 79 72 61 64 6E 6F 63 65 53 003DD P.ADG: .BYTE 21
29 43 41 28 37 21 003DE .ASCII \Secondary days: !7(AC)\
16 003ED
72 74 73 65 72 20 73 73 65 63 63 61 20 6F 4E 003F3 P.ADH: .BYTE 22
69 003F4 .ASCII \No access restrictions\
46 00403
30 30 30 30 30 20 20 20 79 72 61 6D 69 72 50 0040A P.ADI: .BYTE 70
31 31 31 31 31 31 31 31 31 31 30 30 30 30 30 0040B .ASCII \Primary 00000000001111111112222 Seco\
30 30 30 30 30 6F 63 65 53 20 20 32 32 32 32 0041A
32 32 32 32 31 31 31 31 31 31 31 31 31 31 31 31 00429 .ASCII \ndary 00000000001111111112222\
34 33 32 31 30 20 73 72 75 6F 48 20 79 61 44 00433 P.ADJ: .BYTE 70
39 38 37 36 35 34 33 32 31 30 39 38 37 36 35 00442 .ASCII \Day Hours 012345678901234567890123 Day \
38 37 36 35 34 33 32 31 30 20 73 72 75 6F 48 00451
33 32 31 30 39 38 37 36 35 34 33 32 31 30 39 00452
00461
00470
0047A
00489

```



UAF  
VO4  
00F



```
20 20 20 20 20 3A 6D 6C 45 51 54 20 20 4C 55 006D3
78 65 53 57 20 20 4C 55 35 21 006E2
4C 55 39 21 20 3A 74 6E 65 74 006EC
P.ADY: .ASCII \tent: !9UL\
.BYTE 43
.ASCII \CPU: !13AD Enqlm: !5UL Pgflquo: !\
71 6E 45 20 20 44 41 33 31 21 20 3A 55 50 43 006F6
50 20 20 4C 55 35 21 20 20 20 20 20 3A 6D 6C 67 006F7
21 20 20 3A 6F 75 71 6C 66 67 00706
4C 55 39 00715
17 0071F
P.ADZ: .ASCII \9UL\
.BYTE 23
.ASCII \Authorized Privileges: \
76 69 72 50 20 64 65 7A 69 72 6F 68 74 75 41 00722
20 3A 73 65 67 65 6C 69 00723
P.AEA: .BYTE 20
.ASCII \Default Privileges: \
65 6C 69 76 69 72 50 20 74 6C 75 61 66 65 44 0073A
20 3A 73 65 67 0073B
00 0074A
P.AEB: .BYTE 0
0074F .BLKB 0
00750 P.AEC: .BYTE 4
6E 6F 4D 20 00751 .ASCII \ Mon\
04 00755 P.AED: .BYTE 4
65 75 54 20 00756 .ASCII \ Tue\
04 0075A P.AEE: .BYTE 4
64 65 57 20 0075B .ASCII \ Wed\
04 0075F P.AEF: .BYTE 4
75 68 54 20 00760 .ASCII \ Thu\
04 00764 P.AEG: .BYTE 4
69 72 46 20 00765 .ASCII \ Fri\
04 00769 P.AEH: .BYTE 4
74 61 53 20 0076A .ASCII \ Sat\
04 0076E P.AEI: .BYTE 4
6E 75 53 20 0076F .ASCII \ Sun\
04 00773 P.AEJ: .BYTE 4
20 20 20 20 00774 .ASCII \ \
06 00778 P.AEK: .BYTE 6
74 69 64 75 41 20 00779 .ASCII \ Audit\
08 0077F P.AEL: .BYTE 8
65 76 69 74 70 61 43 20 00780 .ASCII \ Captive\
07 00788 P.AEM: .BYTE 7
69 6C 63 66 65 44 20 00789 .ASCII \ Defcli\
08 00790 P.AEN: .BYTE 8
79 6C 74 63 73 69 44 20 00791 .ASCII \ Disctly\
08 00799 P.AEO: .BYTE 8
6C 69 61 6D 73 69 44 20 0079A .ASCII \ Dismail\
08 007A2 P.AEP: .BYTE 11
6C 69 61 6D 77 65 6E 73 69 44 20 007A3 .ASCII \ Disnewmail\
0D 007AE P.AEQ: .BYTE 13
74 63 65 6E 6E 6F 63 65 72 73 69 44 20 007AF .ASCII \ Disreconnect\
0A 007BC P.AER: .BYTE 10
74 72 6F 70 65 72 73 69 44 20 007BD .ASCII \ Disreport\
08 007C7 P.AES: .BYTE 8
72 65 73 75 73 69 44 20 007C8 .ASCII \ Disuser\
08 007D0 P.AET: .BYTE 11
65 6D 6F 63 6C 65 77 73 69 44 20 007D1 .ASCII \ Diswelcome\
07 007DC P.AEU: .BYTE 7
64 77 70 6E 65 47 20 007DD .ASCII \ Genpwd\
08 007E4 P.AEV: .BYTE 8
64 77 70 6B 63 6F 4C 20 007E5 .ASCII \ Lockpwd\
0C 007ED P.AEW: .BYTE 12
```

UAFMAIN  
V04-000

display\_full - writes the full user display

K 4  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 133  
(27)

```

64 65 72 69 70 78 65 5F 64 77 50 20 007EE
65 72 64 65 72 69 70 78 65 5F 32 64 77 50 20 007FA
69 70 78 65 20 65 72 70 28 20 20 20 20 007FB
00 00 00 29 64 00808
00817
```

```

P.AEX: .ASCII \ Pwd_expired\
        .BYTE 13
P.AEY: .ASCII \ Pwd2_expired\
        .ASCII \ (pre-expired)\<0><0><0>
        .
        .
        .
```

.PSECT \$OWNS,NOEXE,2

```

00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 0108C
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 010A4
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 010BC
010C4
```

```

FLAGS_VECTOR:
.ADDRESS P.AEN, P.AEM, P.AEV, P.AEL, P.AES, -
        P.AET, P.AEP, P.AEO, P.AEU, P.AEW, P.AEX, -
        P.AEK, P.AER, P.AEQ
.BLK 72
```

```

USERNAME= P.ACY
ACCOUNT= P.ACZ
CLI_TABLE= P.ADA
DEFAULT= P.ADB
LGICMD= P.ADC
FLAGS= P.ADD
FLAG_PAD= P.ADE
PRIMDAYS= P.ADF
SECDAYS= P.ADG
NORESTRICT= P.ADH
ACCESSHDR1= P.ADI
ACCESSHDR2= P.ADJ
NETACCESS= P.ADK
BATAccess= P.ADL
LOCAccess= P.ADM
DIAAccess= P.ADN
REMAccess= P.ADO
EXPIRATION= P.ADP
PWDDATA= P.ADQ
LASTLOGIN= P.ADR
QUOTA1= P.ADS
QUOTA2= P.ADT
QUOTA3= P.ADU
QUOTA4= P.ADV
QUOTA5= P.ADW
QUOTA6= P.ADX
QUOTA7= P.ADY
PRIVS= P.ADZ
DEFPRIVS= P.AEA
NULLSTR= P.AEB
MON= P.AEC
TUE= P.AED
WED= P.AEE
THU= P.AEF
FRI= P.AEG
SAT= P.AEH
SUN= P.AEI
NODAY= P.AEJ
CPUTIME= RECBUF+556
```

.PSECT \$CODES,NOWRT,2

```
OFFC 00000 DISPLAY_FULL:
Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
3548
5E FF1C CE 9E 00002 MOVAB -228(SP), $P
31 00000000' 00 E9 00007 BLBC STR WILD, 1$
35 00000000' 00 9E 0000E MOVAB MATCH_TOKEN, R5
33 00000000' 00 9E 00015 MOVAB RECBUF+4, R3
20 00000000' 20 3A 0001C LOCC #32, #32, RECBUF+4
32 E0 A0 9E 00024 MOVAB -32(R0), R2
32 CE 00028 MNEGL R2, R2
54 00000000' 00 D0 0002B MOVL MATCH_TOKENLEN, R4
00000000G 00 16 00032 JSB FMG$MATCH_NAME
04 50 E8 00038 BLBS R0, 1$
50 01 D0 0003B MOVL #1, R0
04 0003E RET
00000000' 00 01 DJ 0003F 1$: MOVL #1, FOUND_MATCH
50 00000000' 00 D0 00046 MOVL RABPTR, R0
22 A0 B4 0004D CLRW 34(R0)
50 DD 00050 PUSHL R0
00000000G 00 01 FB 00052 CALLS #1, SYSS$PUT
00000000' 00 00 9B 00059 MOVZBW USERNAME, FAODSC
00000000' 00 00 9E 00064 MOVAB USERNAME+1, FAODSC+4
00000000' 00 00 9F 0006F PUSHAB RECBUF+84
00000000' 00 00 9F 00075 PUSHAB RECBUF+4
20 DD 0007B PUSHL #32
00000000' 00 00 9F 0007D PUSHAB DISDSC
7E 00000000' 00 22 C1 00083 ADDL3 #34, RABPTR, -(SP)
00000000' 00 00 9F 0008B PUSHAB FAODSC
00000000G 00 06 FB 00091 CALLS #6, SYSS$FAO
00000000' 00 00 DD 00098 PUSHL RABPTR
00000000G 00 01 FB 0009E CALLS #1, SYSS$PUT
00000000' 00 00 9B 000A5 MOVZBW ACCOUNT, FAODSC
00000000' 00 00 9E 000B0 MOVAB ACCOUNT+1, FAODSC+4
50 00000000' 00 D0 000B8 MOVL RECBUF+36, R0
50 DD 000C2 PUSHL R0
50 DD 000C4 PUSHL R0
00000000' 00 00 9F 000C6 PUSHAB RECBUF+52
20 DD 000CC PUSHL #32
00000000' 00 00 9F 000CE PUSHAB DISDSC
7E 00000000' 00 22 C1 000D4 ADDL3 #34, RABPTR, -(SP)
00000000' 00 00 9F 000DC PUSHAB FAODSC
00000000G 00 07 FB 000E2 CALLS #7, SYSS$FAO
00000000' 00 00 DD 000E9 PUSHL RABPTR
00000000G 00 01 FB 000EF CALLS #1, SYSS$PUT
00000000' 00 00 9B 000F6 MOVZBW CLI_TABLE, FAODSC
00000000' 00 00 9E 00101 MOVAB CLI_TABLE+1, FAODSC+4
00000000' 00 00 9F 0010C PUSHAB RECBUF+308
00000000' 00 00 9F 00112 PUSHAB RECBUF+276
00000000' 00 00 9F 00118 PUSHAB DISDSC
7E 00000000' 00 22 C1 0011E ADDL3 #34, RABPTR, -(SP)
00000000' 00 00 9F 00126 PUSHAB FAODSC
00000000G 00 05 FB 0012C CALLS #5, SYSS$FAO
00000000' 00 00 DD 00133 PUSHL RABPTR
00000000G 00 01 FB 00139 CALLS #1, SYSS$PUT
00000000' 00 00 9B 00140 MOVZBW DEFAULT, FAODSC
00000000' 00 00 9E 00148 MOVAB DEFAULT+1, FAODSC+4
00000000' 00 00 9F 00156 PUSHAB RECBUF+148
00000000' 00 00 9F 0015C PUSHAB RECBUF+116
3685
3689
3693
```

display\_full - writes the full user display

M 4  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1Page 135  
(27)

7E 00000000'	00	00000000'	00	9F 00162	PUSHAB	DISDSC		
			22	C1 00168	ADDL3	#34, RABPTR, -(SP)		
00000000G	00	00000000'	00	9F 00170	PUSHAB	FAODSC		
			05	FB 00176	CALLS	#5, SYSSFAO		
00000000G	00	00000000'	00	DD 0017D	PUSHL	RABPTR		
00000000'	00	00000000'	01	FB 00183	CALLS	#1, SYSSPUT		
00000000'	00	00000000'	00	9B 0018A	MOVZBW	LGICMD, FAODSC		3696
00000000'	00	00000000'	00	9E 00195	MOVAB	LGICMD+1, FAODSC+4		
			00	9F 001A0	PUSHAB	RECBUF+212		
			00	9F 001A6	PUSHAB	DISDSC		
7E 00000000'	00		22	C1 001AC	ADDL3	#34, RABPTR, -(SP)		
			00	9F 001B4	PUSHAB	FAODSC		
00000000G	00		04	FB 001BA	CALLS	#4, SYSSFAO		
			00	DD 001C1	PUSHL	RABPTR		
00000000G	00		01	FB 001C7	CALLS	#1, SYSSPUT		
			57	D4 001CE	CLRL	COUNT		3698
	5B	00000000'	00	9A 001D0	MOVZBL	FLAGS, R11		3699
	56		5B	D0 001D7	MOVL	R11, LCOUNT		
			59	D4 001DA	CLRL	J		3700
47 00000000'	00		59	E1 001DC	BBC	J, RECBUF+468, 4\$		3703
	5A	00000000'00	49	D0 001E4	MOVL	FLAGS_VECTOR[J], FLAG_STRING		3704
			3D	13 001EC	BEQL	4\$		
	58		6A	9A 001EE	MOVZBL	(FLAG_STRING), R8		3707
	58		56	C0 001F1	ADDL2	LCOUNT, R8		
00000050	8F		58	D1 001F4	CMPL	R8, #80		
			18	1B 001FB	BLEQU	3\$		
	58	00000000'	00	9A 001FD	MOVZBL	FLAG_PAD, R8		3710
44 AE47 00000000'	00		58	28 00204	MOVCL	R8, FLAG_PAD+1, STRING[COUNT]		
	57		58	C0 0020E	ADDL2	R8, COUNT		3711
	56	FE	A8	9E 00211	MOVAB	-2(R8), LCOUNT		3712
	50		6A	9A 00215	MOVZBL	(FLAG_STRING), R0		3714
44 AE47 01	AA		50	28 00218	MOVCL	R0, 1(FLAG_STRING), STRING[COUNT]		
	50		6A	9A 0021F	MOVZBL	(FLAG_STRING), R0		3715
	57		50	C0 00222	ADDL2	R0, COUNT		
	50		6A	9A 00225	MOVZBL	(FLAG_STRING), R0		3716
	56		50	C0 00228	ADDL2	R0, LCOUNT		
AD	59		1F	F3 0022B	AOBLEQ	#31, J, 2\$		3700
00000000'	00		5B	B0 0022F	MOVW	R11, FAODSC		3721
00000000'	00	00000000'	00	9E 00236	MOVAB	FLAGS+1, FAODSC+4		
		44	AE	9F 00241	PUSHAB	STRING		
			57	DD 00244	PUSHL	COUNT		
			00	9F 00246	PUSHAB	DISDSC		
7E 00000000'	00		22	C1 0024C	ADDL3	#34, RABPTR, -(SP)		
			00	9F 00254	PUSHAB	FAODSC		
00000000G	00		05	FB 0025A	CALLS	#5, SYSSFAO		
			00	DD 00261	PUSHL	RABPTR		
00000000G	00		01	FB 00267	CALLS	#1, SYSSPUT		
00000000'	00	00000000'	00	9B 0026E	MOVZBW	PRIMDAYS, FAODSC		3730
00000000'	00	00000000'	00	9E 00279	MOVAB	PRIMDAYS+1, FAODSC+4		
09 00000000'	00		06	E1 00284	BBC	#6, RECBUF+514, 5\$		
	50	00000000'	00	9E 0028C	MOVAB	NODAY, R0		
			07	11 00293	BRB	6\$		
	50	00000000'	00	9E 00295	MOVAB	SUN, R0		
			50	DD 0029C	PUSHL	R0		
09 00000000'	00		05	E1 0029E	BBC	#5, RECBUF+514, 7\$		
	50	00000000'	00	9E 002A6	MOVAB	NODAY, R0		
			07	11 002AD	BRB	8\$		



display\_full - writes the full user display

N 4  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 136  
(27)

	50	00000000'	00	9E	002AF	7\$:	MOVAB	SAT, R0	
			50	DD	002B6	8\$:	PUSHL	R0	
09	00000000'	00	04	E1	002B8		BBC	#4, RECBUF+514, 9\$	
	50	00000000'	00	9E	002C0		MOVAB	NODAY, R0	
			07	11	002C7		BRB	10\$	
	50	00000000'	00	9E	002C9	9\$:	MOVAB	FRI, R0	
			50	DD	002D0	10\$:	PUSHL	R0	
09	00000000'	00	03	E1	002D2		BBC	#3, RECBUF+514, 11\$	
	50	00000000'	00	9E	002DA		MOVAB	NODAY, R0	
			07	11	002E1		BRB	12\$	
	50	00000000'	00	9E	002E3	11\$:	MOVAB	THU, R0	
			50	DD	002EA	12\$:	PUSHL	R0	
09	00000000'	00	02	E1	002EC		BBC	#2, RECBUF+514, 13\$	
	50	00000000'	00	9E	002F4		MOVAB	NODAY, R0	
			07	11	002FB		BRB	14\$	
	50	00000000'	00	9E	002FD	13\$:	MOVAB	WED, R0	
			50	DD	00304	14\$:	PUSHL	R0	
09	00000000'	00	01	E1	00306		BBC	#1, RECBUF+514, 15\$	
	50	00000000'	00	9E	0030E		MOVAB	NODAY, R0	
			07	11	00315		BRB	16\$	
	50	00000000'	00	9E	00317	15\$:	MOVAB	TUE, R0	
			50	DD	0031E	16\$:	PUSHL	R0	
	09	00000000'	00	E9	00320		BLBC	RECBUF+514, 17\$	
	50	00000000'	00	9E	00327		MOVAB	NODAY, R0	
			07	11	0032E		BRB	18\$	
	50	00000000'	00	9E	00330	17\$:	MOVAB	MON, R0	
			50	DD	00337	18\$:	PUSHL	R0	
		00000000'	00	9F	00339		PUSHAB	DISDSC	
7E	00000000'	00	22	C1	0033F		ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F	00347		PUSHAB	FAODSC	
	00000000G	00	0A	FB	0034D		CALLS	#10, SYSSFAO	
		00000000'	00	DD	00354		PUSHL	RABPTR	
	00000000G	00	01	FB	0035A		CALLS	#1, SYSSPUT	
	00000000'	00	00	9B	00361		MOVZBW	SECDAYS, FAODSC	
	00000000'	00	00	9E	0036C		MOVAB	SECDAYS+1, FAODSC+4	
09	00000000'	00	06	E1	00377		BBC	#6, RECBUF+514, 19\$	
	50	00000000'	00	9E	0037F		MOVAB	SUN, R0	
			07	11	00386		BRB	20\$	
	50	00000000'	00	9E	00388	19\$:	MOVAB	NODAY, R0	
			50	DD	0038F	20\$:	PUSHL	R0	
09	00000000'	00	05	E1	00391		BBC	#5, RECBUF+514, 21\$	
	50	00000000'	00	9E	00399		MOVAB	SAT, R0	
			07	11	003A0		BRB	22\$	
	50	00000000'	00	9E	003A2	21\$:	MOVAB	NODAY, R0	
			50	DD	003A9	22\$:	PUSHL	R0	
09	00000000'	00	04	E1	003AB		BBC	#4, RECBUF+514, 23\$	
	50	00000000'	00	9E	003B3		MOVAB	FRI, R0	
			07	11	003BA		BRB	24\$	
	50	00000000'	00	9E	003BC	23\$:	MOVAB	NODAY, R0	
			50	DD	003C3	24\$:	PUSHL	R0	
09	00000000'	00	03	E1	003C5		BBC	#3, RECBUF+514, 25\$	
	50	00000000'	00	9E	003CD		MOVAB	THU, R0	
			07	11	003D4		BRB	26\$	
	50	00000000'	00	9E	003D6	25\$:	MOVAB	NODAY, R0	
			50	DD	003DD	26\$:	PUSHL	R0	
09	00000000'	00	02	E1	003DF		BBC	#2, RECBUF+514, 27\$	
	50	00000000'	00	9E	003E7		MOVAB	WED, R0	

3739



display\_full - writes the full user display

B 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 B11ss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 137  
(27)

		50	00000000'	07	11	003EE	BRB	28\$		
				00	9E	003F0	MOVAB	NODAY, R0		
				50	DD	003F7	PUSHL	R0		
09	00000000'	00		01	E1	003F9	BBC	#1, RECBUF+514, 29\$		
		50	00000000'	00	9E	00401	MOVAB	TUE, R0		
				07	11	00408	BRB	30\$		
		50	00000000'	00	9E	0040A	MOVAB	NODAY, R0		
				50	DD	00411	PUSHL	R0		
		09	00000000'	00	E9	00413	BLBC	RECBUF+514, 31\$		
		50	00000000'	00	9E	0041A	MOVAB	MON, R0		
				07	11	00421	BRB	32\$		
		50	00000000'	00	9E	00423	MOVAB	NODAY, R0		
				50	DD	0042A	PUSHL	R0		
			00000000'	00	9F	0042C	PUSHAB	DISDSC		
7E	00000000'	00		22	C1	00432	ADDL3	#34, RABPTR, -(SP)		
			00000000'	00	9F	0043A	PUSHAB	FAODSC		
		00000000G	00	0A	FB	00440	CALLS	#10, SYSSFAO		
			00000000'	00	DD	00447	PUSHL	RABPTR		
		00000000G	00	01	FB	0044D	CALLS	#1, SYSSPUT		
00000000'	52	00000000'	00	22	C1	00454	ADDL3	#34, RABPTR, R2		3744
	00		1E	00	3B	0045C	SKPC	#0, #30, RECBUF+472		3741
				02	12	00464	BNEQ	33\$		
				51	D4	00466	CLRL	R1		
				51	D5	00468	TSTL	R1		
				3B	12	0046A	BNEQ	34\$		
		00000000'	00	00	9B	0046C	MOVZBW	NORESTRICT, FAODSC		3744
		00000000'	00	00	9E	00477	MOVAB	NORESTRICT+1, FAODSC+4		
			00000000'	00	9F	00482	PUSHAB	DISDSC		
				52	DD	00488	PUSHL	R2		
		00000000'	00	00	9F	0048A	PUSHAB	FAODSC		
		00000000G	00	03	FB	00490	CALLS	#3, SYSSFAO		
			00000000'	00	DD	00497	PUSHL	RABPTR		
		00000000G	00	01	FB	0049D	CALLS	#1, SYSSPUT		
				00D5	31	004A4	BRW	35\$		3741
		00000000'	00	00	9B	004A7	MOVZBW	ACCESSHDR1, FAODSC		3748
		00000000'	00	00	9E	004B2	MOVAB	ACCESSHDR1+1, FAODSC+4		
			00000000'	00	9F	004BD	PUSHAB	DISDSC		
				52	DD	004C3	PUSHL	R2		
		00000000'	00	00	9F	004C5	PUSHAB	FAODSC		
		00000000G	00	03	FB	004CB	CALLS	#3, SYSSFAO		
			00000000'	00	DD	004D2	PUSHL	RABPTR		
		00000000G	00	01	FB	004D8	CALLS	#1, SYSSPUT		
		00000000'	00	00	9B	004DF	MOVZBW	ACCESSHDR2, FAODSC		3749
		00000000'	00	00	9E	004EA	MOVAB	ACCESSHDR2+1, FAODSC+4		
			00000000'	00	9F	004F5	PUSHAB	DISDSC		
7E	00000000'	00		22	C1	004FB	ADDL3	#34, RABPTR, -(SP)		
			00000000'	00	9F	00503	PUSHAB	FAODSC		
		00000000G	00	03	FB	00509	CALLS	#3, SYSSFAO		
			00000000'	00	DD	00510	PUSHL	RABPTR		
		00000000G	00	01	FB	00516	CALLS	#1, SYSSPUT		
			00000000'	00	9F	0051D	PUSHAB	RECBUF+472		3751
			00000000'	00	9F	00523	PUSHAB	NETACCESS		
		00000000V	00	02	FB	00529	CALLS	#2, DISPLAY_HOURS		
			00000000'	00	9F	00530	PUSHAB	RECBUF+478		3752
			00000000'	00	9F	00536	PUSHAB	BATACCESS		
		00000000V	00	02	FB	0053C	CALLS	#2, DISPLAY_HOURS		
			00000000'	00	9F	00543	PUSHAB	RECBUF+484		3753

display\_full - writes the full user display

C 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1Page 138  
(27)

00000000V	00	00000000'	00	9F 00549	PUSHAB	LOCACCESS		
		00000000'	02	FB 0054F	CALLS	#2, DISPLAY_HOURS		
		00000000'	00	9F 00556	PUSHAB	RECBUF+490	3754	
00000000V	00	00000000'	00	9F 0055C	PUSHAB	DIAACCESS		
		00000000'	02	FB 00562	CALLS	#2, DISPLAY_HOURS		
		00000000'	00	9F 00569	PUSHAB	RECBUF+496	3755	
00000000V	00	00000000'	00	9F 0056F	PUSHAB	REMACCESS		
		28	02	FB 00575	CALLS	#2, DISPLAY_HOURS		
			AE	9F 0057C	PUSHAB	TIME1	3758	
			11	DD 0057F	PUSHL	#17		
		00000000'	00	9F 00581	PUSHAB	RECBUF+364		
00000000V	00	00000000'	03	FB 00587	CALLS	#3, CONVERT_TIME		
00000000'	00	00000000'	00	9B 0058E	MOVZBW	EXPIRATION, FAODSC	3762	
00000000'	00	00000000'	00	9E 00599	MOVAB	EXPIRATION+1, FAODSC+4		
	7E	00000000'	00	3C 005A4	MOVZWL	RECBUF+356, -(SP)		
	7E	00000000'	00	9A 005AB	MOVZBL	RECBUF+362, -(SP)		
		30	AE	9F 005B2	PUSHAB	TIME1		
			11	DD 005B5	PUSHL	#17		
		00000000'	00	9F 005B7	PUSHAB	DISDSC		
7E 00000000'	00	00000000'	22	C1 005BD	ADDL3	#34, RABPTR, -(SP)		
		00000000'	00	9F 005C5	PUSHAB	FAODSC		
00000000G	00	00000000'	07	FB 005CB	CALLS	#7, SYSSFAO		
		00000000'	00	DD 005D2	PUSHL	RABPTR		
00000000G	00	00000000'	01	FB 005D8	CALLS	#1, SYSSPUT		
		28	AE	9F 005DF	PUSHAB	TIME1	3764	
			0A	DD 005E2	PUSHL	#10		
		00000000'	00	9F 005E4	PUSHAB	RECBUF+372		
00000000V	00	00000000'	03	FB 005EA	CALLS	#3, CONVERT_TIME		
	50	00000000'	00	9E 005F1	MOVAB	RECBUF+380, PTR	3765	
FFFFFFFF	8F		60	D1 005F8	CMPL	(PTR), #-1	3766	
			15	12 005FF	BNEQ	368		
FFFFFFFF	8F	04	A0	D1 00601	CMPL	4(PTR), #-1		
			0B	12 00609	BNEQ	368		
14 AE 00000000'	00		11	28 0060B	MOVC3	#17, P.AEY, TIME2	3767	
			0E	11 00614	BRB	378		
		14	AE	9F 00616	PUSHAB	TIME2	3769	
			11	DD 00619	PUSHL	#17		
			50	DD 0061B	PUSHL	PTR		
00000000V	00		03	FB 0061D	CALLS	#3, CONVERT_TIME		
			5E	DD 00624	PUSHL	SP	3770	
			11	DD 00626	PUSHL	#17		
		00000000'	00	9F 00628	PUSHAB	RECBUF+388		
00000000V	00	00000000'	03	FB 0062E	CALLS	#3, CONVERT_TIME		
00000000'	00	00000000'	00	9B 00635	MOVZBW	PWDDATA, FAODSC	3776	
00000000'	00	00000000'	00	9E 00640	MOVAB	PWDDATA+1, FAODSC+4		
			5E	DD 0064B	PUSHL	SP		
50 00000000'	00	00000000'	00	C9 0064D	BISL3	RECBUF+392, RECBUF+388, R0		
			04	12 00659	BNEQ	388		
			7E	D4 0065B	CLRL	-(SP)		
			02	11 0065D	BRB	398		
			11	DD 0065F	PUSHL	#17	388:	
		1C	AE	9F 00661	PUSHAB	TIME2	398:	
			11	DD 00664	PUSHL	#17		
		38	AE	9F 00666	PUSHAB	TIME1		
			0A	DD 00669	PUSHL	#10		
		00000000'	00	9F 0066B	PUSHAB	DISDSC		
7E 00000000'	00		22	C1 00671	ADDL3	#34, RABPTR, -(SP)		

	00000000G	00	00000000'	00	9F 00679	PUSHAB	FAODSC		
				09	FB 0067F	CALLS	#9, SYSSFAO		
	00000000G	00	00000000'	00	DD 00686	PUSHL	RABPTR		
				01	FB 0068C	CALLS	#1, SYSSPUT		
			28	AE	9F 00693	PUSHAB	TIME1	3778	
				11	DD 00696	PUSHL	#17		
	00000000V	00	00000000'	00	9F 00698	PUSHAB	RECBUF+396		
			14	AE	9F 0069E	CALLS	#3, CONVERT_TIME	3779	
				11	DD 006A8	PUSHL	#17		
			00000000'	00	9F 006AA	PUSHAB	RECBUF+404		
	00000000V	00	00000000'	03	FB 006B0	CALLS	#3, CONVERT_TIME		
	00000000'	00	00000000'	00	9B 006B7	MOVZBW	LASTLOGIN, FAODSC	3782	
	00000000'	00	00000000'	00	9E 006C2	MOVAB	LASTLOGIN+1, FAODSC+4		
			14	AE	9F 006CD	PUSHAB	TIME2		
				11	DD 006D0	PUSHL	#17		
			30	AE	9F 006D2	PUSHAB	TIME1		
				11	DD 006D5	PUSHL	#17		
			00000000'	00	9F 006D7	PUSHAB	DISDSC		
7E	00000000'	00	00000000'	22	C1 006DD	ADDL3	#34, RABPTR, -(SP)		
			00000000'	00	9F 006E5	PUSHAB	FAODSC		
	00000000G	00	00000000'	07	FB 006EB	CALLS	#7, SYSSFAO		
			00000000'	00	DD 006F2	PUSHL	RABPTR		
51	00000000'	00	00000000G	00	01 FB 006F8	CALLS	#1, SYSSPUT		
		1F		01	EF 006FF	EXTZV	#1, #31, CPUTIME, R1	3784	
		09	00000000'	00	E9 00708	BLBC	CPUTIME, 408	3785	
		50	FFFE7960	8F	D0 0070F	MOVL	#-100000, R0		
				02	11 00716	BRB	418		
				50	D4 00718	CLRL	R0		
3C	AE	50	51 FFFCF2C0	8F	7A 0071A	EMUL	#-200000, R1, R0, DELTA_TIME	3784	
			28	AE	9F 00724	PUSHAB	TIME1	3786	
				0D	DD 00727	PUSHL	#13		
			44	AE	9F 00729	PUSHAB	DELTA_TIME		
	00000000V	00	00000000'	03	FB 0072C	CALLS	#3, CONVERT_TIME		
	00000000'	00	00000000'	00	9B 00733	MOVZBW	QUOTA1, FAODSC	3791	
	00000000'	00	00000000'	00	9E 0073E	MOVAB	QUOTA1+1, FAODSC+4		
			00000000'	00	DD 00749	PUSHL	RECBUF+560		
		7E	00000000'	00	3C 0074F	MOVZWL	RECBUF+536, -(SP)		
		7E	00000000'	00	3C 00756	MOVZWL	RECBUF+518, -(SP)		
			00000000'	00	9F 0075D	PUSHAB	DISDSC		
7E	00000000'	00	00000000'	22	C1 00763	ADDL3	#34, RABPTR, -(SP)		
			00000000'	00	9F 0076B	PUSHAB	FAODSC		
	00000000G	00	00000000'	06	FB 00771	CALLS	#6, SYSSFAO		
			00000000'	00	DD 00778	PUSHL	RABPTR		
	00000000G	00	00000000'	01	FB 0077E	CALLS	#1, SYSSPUT		
	00000000'	00	00000000'	00	9B 00785	MOVZBW	QUOTA2, FAODSC	3796	
	00000000'	00	00000000'	00	9E 00790	MOVAB	QUOTA2+1, FAODSC+4		
			00000000'	00	DD 00798	PUSHL	RECBUF+564		
		7E	00000000'	00	3C 007A1	MOVZWL	RECBUF+538, -(SP)		
		7E	00000000'	00	3C 007A8	MOVZWL	RECBUF+520, -(SP)		
			00000000'	00	9F 007AF	PUSHAB	DISDSC		
7E	00000000'	00	00000000'	22	C1 007B5	ADDL3	#34, RABPTR, -(SP)		
			00000000'	00	9F 007BD	PUSHAB	FAODSC		
	00000000G	00	00000000'	06	FB 007C3	CALLS	#6, SYSSFAO		
			00000000'	00	DD 007CA	PUSHL	RABPTR		
	00000000G	00	00000000'	01	FB 007D0	CALLS	#1, SYSSPUT		
	00000000'	00	00000000'	00	9B 007D7	MOVZBW	QUOTA3, FAODSC	3801	



display\_full - writes the full user display

E 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

00000000'	00	00000000'	00	9E 007E2	MOVAB	QUOTA3+1, FAODSC+4	
		00000000'	00	DD 007ED	PUSHL	RECBUF+568	
	7E	00000000'	00	3C 007F3	MOVZWL	RECBUF+526, -(SP)	
	7E	00000000'	00	3C 007FA	MOVZWL	RECBUF+522, -(SP)	
		00000000'	00	9F 00801	PUSHAB	DISDSC	
7E 00000000'	00	00000000'	22	C1 00807	ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F 0080F	PUSHAB	FAODSC	
00000000G	00	00000000'	06	FB 00815	CALLS	#6, SYSSFAO	
		00000000'	00	DD 0081C	PUSHL	RABPTR	
00000000G	00	00000000'	01	FB 00822	CALLS	#1, SYSSPUT	
00000000'	00	00000000'	00	9B 00829	MOVZBW	QUOTA4, FAODSC	3806
00000000'	00	00000000'	00	9E 00834	MOVAB	QUOTA4+1, FAODSC+4	
		00000000'	00	DD 0083F	PUSHL	RECBUF+544	
	7E	00000000'	00	3C 00845	MOVZWL	RECBUF+528, -(SP)	
	7E	00000000'	00	3C 0084C	MOVZWL	RECBUF+524, -(SP)	
		00000000'	00	9F 00853	PUSHAB	DISDSC	
7E 00000000'	00	00000000'	22	C1 00859	ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F 00861	PUSHAB	FAODSC	
00000000G	00	00000000'	06	FB 00867	CALLS	#6, SYSSFAO	
		00000000'	00	DD 0086E	PUSHL	RABPTR	
00000000G	00	00000000'	01	FB 00874	CALLS	#1, SYSSPUT	
00000000'	00	00000000'	00	9B 0087B	MOVZBW	QUOTA5, FAODSC	3811
00000000'	00	00000000'	00	9E 00886	MOVAB	QUOTA5+1, FAODSC+4	
		00000000'	00	DD 00891	PUSHL	RECBUF+540	
	7E	00000000'	00	3C 00897	MOVZWL	RECBUF+532, -(SP)	
	7E	00000000'	00	9A 0089E	MOVZBL	RECBUF+516, -(SP)	
		00000000'	00	9F 008A5	PUSHAB	DISDSC	
7E 00000000'	00	00000000'	22	C1 008AB	ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F 008B3	PUSHAB	FAODSC	
00000000G	00	00000000'	06	FB 008B9	CALLS	#6, SYSSFAO	
		00000000'	00	DD 008C0	PUSHL	RABPTR	
00000000G	00	00000000'	01	FB 008C6	CALLS	#1, SYSSPUT	
00000000'	00	00000000'	00	9B 008CD	MOVZBW	QUOTA6, FAODSC	3816
00000000'	00	00000000'	00	9E 008D8	MOVAB	QUOTA6+1, FAODSC+4	
		00000000'	00	DD 008E3	PUSHL	RECBUF+548	
	7E	00000000'	00	3C 008E9	MOVZWL	RECBUF+530, -(SP)	
	7E	00000000'	00	9A 008F0	MOVZBL	RECBUF+517, -(SP)	
		00000000'	00	9F 008F7	PUSHAB	DISDSC	
7E 00000000'	00	00000000'	22	C1 008FD	ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F 00905	PUSHAB	FAODSC	
00000000G	00	00000000'	06	FB 0090B	CALLS	#6, SYSSFAO	
		00000000'	00	DD 00912	PUSHL	RABPTR	
00000000G	00	00000000'	01	FB 00918	CALLS	#1, SYSSPUT	
00000000'	00	00000000'	00	9B 0091F	MOVZBW	QUOTA7, FAODSC	3821
00000000'	00	00000000'	00	9E 0092A	MOVAB	QUOTA7+1, FAODSC+4	
		00000000'	00	DD 00935	PUSHL	RECBUF+552	
	7E	00000000'	00	3C 0093B	MOVZWL	RECBUF+534, -(SP)	
		30	AE	9F 00942	PUSHAB	TIME1	
			0D	DD 00945	PUSHL	#13	
		00000000'	00	9F 00947	PUSHAB	DISDSC	
7E 00000000'	00	00000000'	22	C1 0094D	ADDL3	#34, RABPTR, -(SP)	
		00000000'	00	9F 00955	PUSHAB	FAODSC	
00000000G	00	00000000'	07	FB 0095B	CALLS	#7, SYSSFAO	
		00000000'	00	DD 00962	PUSHL	RABPTR	
00000000G	00	00000000'	01	FB 00968	CALLS	#1, SYSSPUT	
00000000'	00	00000000'	00	9B 0096F	MOVZBW	PRIVS, FAODSC	3823
00000000'	00	00000000'	00	9E 0097A	MOVAB	PRIVS+1, FAODSC+4	

display\_full - writes the full user display

F 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 B11ss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 Page 141  
(27)

7E 00000000'	00	00000000'	00	9F 00985	PUSHAB	DISDSC		
			22	C1 0098B	ADDL3	#34, RABPTR, -(SP)		
00000000G	00	00000000'	00	9F 00993	PUSHAB	FAODSC		
			03	FB 00999	CALLS	#3, SYSSFA0		
00000000G	00	00000000'	00	DD 009A0	PUSHL	RABPTR		
			01	FB 009A6	CALLS	#1, SYSSPUT		
00000000V	00	00000000'	00	9F 009AD	PUSHAB	RECBUF+412		3824
			01	FB 009B3	CALLS	#1, PRINT_PRIV		
00000000'	00	00000000'	00	9B 009BA	MOVZBW	DEFPRIVS, FAODSC		3826
00000000'	00	00000000'	00	9E 009C5	MOVAB	DEFPRIVS+1, FAODSC+4		
		00000000'	00	9F 009D0	PUSHAB	DISDSC		
7E 00000000'	00		22	C1 009D6	ADDL3	#34, RABPTR, -(SP)		
		00000000'	00	9F 009DE	PUSHAB	FAODSC		
00000000G	00		03	FB 009E4	CALLS	#3, SYSSFA0		
		00000000'	00	DD 009EB	PUSHL	RABPTR		
00000000G	00		01	FB 009F1	CALLS	#1, SYSSPUT		
		00000000'	00	9F 009F8	PUSHAB	RECBUF+420		3827
00000000V	00		01	FB 009FE	CALLS	#1, PRINT_PRIV		
	1B	00000000'	00	E9 00A05	BLBC	RDB_EXISTS, 42\$		3833
00000000G	00		00	FB 00A0C	CALLS	#0, UAF\$BUILD HOLDER		3836
00000000G	00		01	90 00A13	MOVB	#1, RDB_HEADER_FLAG		3837
		00000000G	00	9F 00A1A	PUSHAB	HOLDER		3838
00000000G	00		01	FB 00A20	CALLS	#1, UAF\$WRITE_RIGHTS		
			04	00A27 42\$:	RET			3842

; Routine Size: 2600 bytes, Routine Base: \$CODE\$ + 16CD



display\_hours - display hourly restrictions

```
3775 3843 1 %sbttl 'display_hours - display hourly restrictions'
3776 3844 1 routine display_hours (format_string, hour_vector) : novalue =
3777 3845 2 begin
3778 3846 3
3779 3847 4 ++
3780 3848 5
3781 3849 6 FUNCTIONAL DESCRIPTION:
3782 3850 7
3783 3851 8     Displays the hourly access for primary and secondary days
3784 3852 9     for one access type.
3785 3853 10
3786 3854 11 INPUTS:
3787 3855 12
3788 3856 13     format_string: address of FAO string for display
3789 3857 14     hour_vector: address of UAF record hourly vector
3790 3858 15     RABPTR - RMS data structure for the file
3791 3859 16
3792 3860 17 IMPLICIT INPUTS:
3793 3861 18
3794 3862 19     none
3795 3863 20
3796 3864 21 OUTPUTS:
3797 3865 22
3798 3866 23     none
3799 3867 24
3800 3868 25 IMPLICIT OUTPUTS:
3801 3869 26
3802 3870 27     none
3803 3871 28
3804 3872 29 ROUTINE VALUE:
3805 3873 30
3806 3874 31     none
3807 3875 32
3808 3876 33 SIDE EFFECTS:
3809 3877 34
3810 3878 35     none
3811 3879 36 --
3812 3880 37
3813 3881 38
3814 3882 39 Display strings.
3815 3883 40
3816 3884 41
3817 3885 42 map
3818 3886 43     hour_vector : ref bitvector;
3819 3887 44
3820 3888 45 literal
3821 3889 46     yeschar      = '#';
3822 3890 47     nochar       = '-';
3823 3891 48
3824 3892 49 bind
3825 3893 50     noaccess     = cstring ('----- No access -----');
3826 3894 51     fullaccess   = cstring ('##### Full access #####');
3827 3895 52
3828 3896 53 local
3829 3897 54     pri_access   : vector [24, byte];    ! Character string for primary access
3830 3898 55     sec_access   : vector [24, byte];    ! Character string for secondary access
3831 3899 56
```

display\_hours - display hourly restrictions

```
3832 3900 2 if . (hour_vector[0])<0,24> eql 0
3833 3901 then ch$move (24, fullaccess+1, pri_access)
3834 3902 else if . (hour_vector[0])<0,24> eql 'x'fffff'
3835 3903 then ch$move (24, noaccess+1, pri_access)
3836 3904 else incr j from 0 to 23
3837 3905 do
3838 3906 begin
3839 3907 if .hour_vector[.]
3840 3908 then pri_access[.] = nochar
3841 3909 else pri_access[.] = yeschar;
3842 3910 end;
3843 3911
3844 3912 if . (hour_vector[0])<24,24> eql 0
3845 3913 then ch$move (24, fullaccess+1, sec_access)
3846 3914 else if . (hour_vector[0])<24,24> eql 'x'fffff'
3847 3915 then ch$move (24, noaccess+1, sec_access)
3848 3916 else incr j from 0 to 23
3849 3917 do
3850 3918 begin
3851 3919 if .hour_vector[.]+24]
3852 3920 then sec_access[.] = nochar
3853 3921 else sec_access[.] = yeschar;
3854 3922 end;
3855 3923
3856 3924 faomac (.format_string,
3857 3925 24, pri_access,
3858 3926 24, sec_access
3859 3927 );
3860 3928 end;
```

```
73 65 63 63 61 20 6F 4E 20 20 2D 2D 2D 2D 2D 18 0081C P.AEZ: .BYTE 24
2D 2D 2D 2D 2D 2D 2D 2D 73 0081D .ASCII \----- No access -----\
65 63 63 61 20 6C 6C 75 46 20 23 23 23 23 23 18 00835 P.AFA: .BYTE 24
23 23 23 23 23 23 20 73 73 00836 .ASCII \##### Full access #####\
00845
```

NOACCESS= P.AEZ  
FULLACCESS= P.AFA

```
01FC 00000 DISPLAY_HOURS:
58 00000000' 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8
57 00000000' 00 9E 00009 MOVAB FULLACCESS+1, R8
5E 30 C2 00010 MOVAB FAODSC, R7
56 08 AC D0 00013 SUBL2 #48, SP
18 00 ED 00017 MOVL HOUR_VECTOR, R6
07 12 0001C CMPZV #0, #24, (R6), #0
18 28 0001E BNEQ 1$
29 11 00023 MOVCS #24, FULLACCESS+1, PRI_ACCESS
BRB 6$
```

3844  
3900  
3901

UAFMAIN  
V04-000

display\_hours - display hourly restrictions

1 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 144  
(28)

00FFFFFF	8F	66	18	00	ED	00025	1\$:	CMPZV	#0, #24, (R6), #16777215	3902
				08	12	0002E		BNEQ	2\$	3903
	18	AE	E7	A8	18	28	00030	MOVC3	#24, NOACCESS+1, PRI_ACCESS	3904
				16	11	00036		BRB	6\$	3907
				50	D4	00038	2\$:	CLRL	J	3908
	07		66	50	E1	0003A	3\$:	BBC	J (R6), 4\$	3909
			18	AE40	20	90	0003E	MOVB	#45, PRI_ACCESS[EJ]	3904
					05	11	00043	BRB	5\$	3907
			18	AE40	23	90	00045	MOVB	#35, PRI_ACCESS[EJ]	3908
	EC		50	17	F3	0004A	4\$:	AOBLEQ	#23, J, 3\$	3909
50	66		18	18	EF	0004E	5\$:	EXTZV	#24, #24, (R6), R0	3904
				06	12	00053	6\$:	BNEQ	7\$	3912
	6E		68	18	28	00055		MOVC3	#24, FULLACCESS+1, SEC_ACCESS	3913
				28	11	00059		BRB	12\$	3914
		00FFFFFF	8F	50	D1	0005B	7\$:	CMPL	R0, #16777215	3915
	6E		E7	A8	07	12	00062	BNEQ	8\$	3916
				18	28	00064		MOVC3	#24, NOACCESS+1, SEC_ACCESS	3919
				18	11	00069		BRB	12\$	3920
			51	50	D4	0006B	8\$:	CLRL	J	3921
	06		66	A0	9E	0006D	9\$:	MOVAB	24(R0), R1	3927
			6E40	51	E1	00071		BBC	R1, (R6), 10\$	
				20	90	00075		MOVB	#45, SEC_ACCESS[EJ]	3921
			6E40	04	11	00079		BRB	11\$	3916
				23	90	0007B	10\$:	MOVB	#35, SEC_ACCESS[EJ]	3927
	EA		50	17	F3	0007F	11\$:	AOBLEQ	#23, J, 9\$	
			67	04	BC	9B	00083	MOVZBW	@FORMAT_STRING, FAODSC	
04	A7	04	AC	01	C1	00087	12\$:	ADDL3	#1, FORMAT_STRING, FAODSC+4	
				5E	DD	0008D		PUSHL	SP	
				18	DD	0008F		PUSHL	#24	
				20	AE	9F	00091	PUSHAB	PRI_ACCESS	
				18	DD	00094		PUSHL	#24	
	7E	08	A7	F8	A7	9F	00096	PUSHAB	DISDSC	
				22	C1	00099		ADDL3	#34, RABPTR, -(SP)	
		00000000G	00	57	DD	0009E		PUSHL	R7	
				07	FB	000A0		CALLS	#7, SYSSFAO	
		00000000G	00	08	A7	DD	000A7	PUSHL	RABPTR	
				01	FB	000AA		CALLS	#1, SYSSPUT	
				04	00	000B1		RET		3928

; Routine Size: 178 bytes, Routine Base: \$CODE\$ + 20F5

convert\_time - convert time value to string

```
3862 3929 1 %sbttl 'convert_time - convert time value to string'
3863 3930 routine convert_time (time, length, buffer) : novalue =
3864 3931 begin
3865 3932
3866 3933 ++
3867 3934
3868 3935 FUNCTIONAL DESCRIPTION:
3869 3936
3870 3937 Converts the binary time value into a string, substituting
3871 3938 the string " (none)" if the value is zero.
3872 3939
3873 3940 INPUTS:
3874 3941
3875 3942 time: address of quadword time
3876 3943 length: length of output string
3877 3944 buffer: address of output buffer
3878 3945
3879 3946 IMPLICIT INPUTS:
3880 3947
3881 3948 none
3882 3949
3883 3950 OUTPUTS:
3884 3951
3885 3952 none
3886 3953
3887 3954 IMPLICIT OUTPUTS:
3888 3955
3889 3956 none
3890 3957
3891 3958 ROUTINE VALUE:
3892 3959
3893 3960 none
3894 3961
3895 3962 SIDE EFFECTS:
3896 3963
3897 3964 none
3898 3965 --
3899 3966
3900 3967 map
3901 3968 time : ref vector;
3902 3969
3903 3970 local
3904 3971 string_desc : vector [2]; ! descriptor for buffer
3905 3972
3906 3973 if (.time[0] or .time[1]) eql 0
3907 3974 then
3908 3975 begin
3909 3976 ch$fill (' ', .length-6, .buffer);
3910 3977 ch$move (6, uplit byte ('(none)'), .buffer+.length-6);
3911 3978 end
3912 3979
3913 3980 else
3914 3981 begin
3915 3982 string_desc[0] = .length;
3916 3983 string_desc[1] = .buffer;
3917 3984 $asctim (timadr = .time, timbuf = string_desc);
3918 3985 end;
```

: 3919

```
convert_time - convert time value to string
3986 1 end:
```

K 5  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742 Page 146  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 (29)

```
.PSECT SPLITS,NOWRT,NOEXE,2
```

29 65 6E 6F 6E 28 0084E P.AFB: .ASCII \ (none) \

.EXTRN SYSSASCTIM

.PSECT SCODES,NOWRT,2

007C 00000 CONVERT\_TIME:

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

; Routine Size: 65 bytes, Routine Base: \$CODES + 21A7



print\_priv - print privilege bits

```
3921 3987 1 %sbttl 'print_priv - print privilege bits'
3922 3988 1 routine print_priv (prvadr) : novalue =
3923 3989 begin
3924 3990
3925 3991 ++
3926 3992
3927 3993 FUNCTIONAL DESCRIPTION:
3928 3994
3929 3995     Routine to output the names of the privilege bits set
3930 3996     in the privilege vector supplied.
3931 3997
3932 3998 INPUTS:
3933 3999
3934 4000     PRVADR - Address of the privilege vector
3935 4001
3936 4002 IMPLICIT INPUTS:
3937 4003
3938 4004     PRV$AB_NAMES - table of privilege names and bit numbers
3939 4005
3940 4006 OUTPUTS:
3941 4007
3942 4008     none
3943 4009
3944 4010 IMPLICIT OUTPUTS:
3945 4011
3946 4012     none
3947 4013
3948 4014 ROUTINE VALUE:
3949 4015
3950 4016     none
3951 4017
3952 4018 SIDE EFFECTS:
3953 4019
3954 4020     none
3955 4021
3956 4022 --
3957 4023
3958 4024 local
3959 4025     pointer,           ! current location in PRV$AB_NAMES
3960 4026     prvcnt,           ! number of names in DISBUF
3961 4027     symlen,           ! length of bit name string
3962 4028     symmin,           ! minimum symbol length
3963 4029     symval;           ! value (bit number)
3964 4030
3965 4031
3966 4032     Initialize the buffer.
3967 4033
3968 4034
3969 4035     disbuf = ' ';      ! insert blank at start
3970 4036     prvcnt = 0;
3971 4037     rabptr[rab$w_rsz] = 1;
3972 4038     pointer = prv$ab_names; ! point to symbol name table
3973 4039
3974 4040 while (symmin = . (.pointer)<0,8>) neq 0 ! pick up min symbol size
3975 4041 do
3976 4042     begin
3977 4043
```

print\_priv - print privilege bits

```

3978 4044  Pick up the next bit name and number.  If the bit is set, insert
3979 4045  the bit name into the buffer.  When the buffer fills up output them
3980 4046  to the user.
3981 4047
3982 4048
3983 4049
3984 4050      pointer = .pointer + 1;
3985 4051      symval = . (.pointer)<0,8>;      ! get bit number
3986 4052      pointer = .pointer + 1;
3987 4053      symlen = . (.pointer)<0,8>;      ! get name string length
3988 4054      pointer = .pointer + 1;      ! point to string
3989 4055      if . (.prvadr)<.symval,1>
3990 4056      AND CH$NEQ(.SYMLN, .POINTER, 7, UPLIT('UPGRADE'))      !**
3991 4057      AND CH$NEQ(.SYMLN, .POINTER, 9, UPLIT('DOWNGRADE'))      !**
3992 4058      AND CH$NEQ(.SYMLN, .POINTER, 6, UPLIT('TMPJNL'))      !**
3993 4059      AND CH$NEQ(.SYMLN, .POINTER, 6, UPLIT('PRMJNL'))      !**
3994 4060      then
3995 4061      begin
3996 4062
3997 4063
3998 4064      Bit is set.  See if there's room in the buffer and insert it if so,
3999 4065      else output the buffer and start from scratch.
4000 4066
4001 4067
4002 4068      if .rabptr[rab$w_rsz] + .symlen geq 64
4003 4069      then
4004 4070      begin
4005 4071      $put (rab = .rabptr);
4006 4072      prvcnt = 0;
4007 4073      rabptr[rab$w_rsz] = 1;
4008 4074      end;
4009 4075
4010 4076
4011 4077      Insert a blank and append symbol name.
4012 4078
4013 4079
4014 4080      disbuf[.rabptr[rab$w_rsz]] = %char (' ');
4015 4081      rabptr[rab$w_rsz] = .rabptr[rab$w_rsz] + 1;
4016 4082      ch$move (.symlen, .pointer, disbuf[.rabptr[rab$w_rsz]]);
4017 4083      rabptr[rab$w_rsz] = .rabptr[rab$w_rsz] + .symlen;
4018 4084      prvcnt = .prvcnt + 1;      ! one more name in buffer
4019 4085      end;
4020 4086
4021 4087      pointer = .pointer + .symlen;      ! update table pointer over name
4022 4088      end;
4023 4089
4024 4090
4025 4091      Table used up.  If anything is in the buffer, print it.
4026 4092
4027 4093
4028 4094      if .prvcnt gtr 0
4029 4095      then $put (rab = .rabptr);
4030 4096
4031 4097      end;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

00	00	00	45	00	45	44	41	52	47	50	55	00854	P.AFC:	.ASCII	\UPGRADE\<0>	:
				44	41	52	47	4E	57	4F	44	0085C	P.AFD:	.ASCII	\DOWNGRADE\<0><0><0>	:
				00	00	4C	4E	4A	50	4D	54	00868	P.AFE:	.ASCII	\TMPJNL\<0><0>	:
				00	00	4C	4E	4A	4D	52	50	00870	P.AFF:	.ASCII	\PRMJNL\<0><0>	:

.PSECT \$CODE\$,NOWRT,2

				OFFC	00000	PRINT_PRIV:				
			00000000'	00	20	D0	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	3988
					59	D4	00009	MOVL	#32,DISBUF	4035
								CLRL	PRVCNT	4036
		22	50 00000000'	00	D0	00008	MOVL	RABPTR, R0		4037
			A0	01	B0	00012	MOVW	#1,34(R0)		
			57 00000000G	00	9E	00016	MOVAB	PRVSAB NAMES, POINTER		4038
			58	67	9A	0001D	1\$: MOVZBL	(POINTER), SYMMIN		4040
					03	12	00020	BNEQ	2\$	
					0098	31	00022	BRW	6\$	
					57	D6	00025	2\$: INCL	POINTER	4050
			5A	87	9A	00027	MOVZBL	(POINTER)+, SYMVAL		4051
			58	87	9A	0002A	MOVZBL	(POINTER)+, SYMLEN		4053
	03	0:	BC	5A	E0	0002D	BBS	SYMVAL, @PRVADR, 3\$		4055
					0082	31	00032	BRW	5\$	
07	00		67	58	2D	00035	3\$: CMPC5	SYMLEN, (POINTER), #0, #7, P.AFC		4056
					00		0003A			
			00000000'		76	13	0003F	BEQL	5\$	
09	00		67	58	2D	00041	CMPC5	SYMLEN, (POINTER), #0, #9, P.AFD		4057
					00		00046			
06	00		67	6A	13	0004B	BEQL	5\$		
				58	2D	0004D	CMPC5	SYMLEN, (POINTER), #0, #6, P.AFE		4058
			00000000'		00		00052			
				5E	13	00057	BEQL	5\$		
06	00		67	58	2D	00059	CMPC5	SYMLEN, (POINTER), #0, #6, P.AFF		4059
			00000000'		00		0005E			
				52	13	00063	BEQL	5\$		
			50 00000000'	00	D0	00065	MOVL	RABPTR, R0		4068
			51	22	A0	3C	0006C	MOVZWL	34(R0), R1	
			51		58	C0	00070	ADDL2	SYMLEN, R1	
			3F		51	D1	00073	CMPL	R1, #63	
				16	15	00076	BLEQ	4\$		
				50	DD	00078	PUSHL	R0		4071
			00000000G	00	01	FB	0007A	CALLS	#1,SYSPUT	
				59	D4	00081	CLRL	PRVCNT		4072
			50 00000000'	00	D0	00083	MOVL	RABPTR, R0		4073
			22	A0	B0	0008A	MOVW	#1,34(R0)		
			50 00000000'	00	D0	0008E	4\$: MOVL	RABPTR, R0		4080
			56	22	A0	9E	00095	MOVAB	34(R0), R6	
			50		66	3C	00099	MOVZWL	(R6), R0	
			00000000'0040		20	90	0009C	MOVB	#32,DISBUF[R0]	
				66	B6	000A4	INCL	(R6)		4081
				50	66	3C	000A6	MOVZWL	(R6), R0	4082
			00000000'0040	67	58	28	000A9	MOVC3	SYMLEN, (POINTER), DISBUF[R0]	
				66	58	A0	000B2	ADDW2	SYMLEN, (R6)	4083
					59	D6	000B5	INCL	PRVCNT	4084

UAFMAIN  
V04-000

print\_priv - print privilege bits

B 6  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 150  
(30)

57	58	C0	000B7	5\$:	ADDL2	SYMLEN, POINTER	:	4087
	FF60	31	000BA		BRW	1\$	:	4040
	59	D5	000BD	6\$:	TSTL	PRVCNT	:	4094
	0D	15	000BF		BLEQ	7\$	:	
00000000G	00	00	DD	000C1	PUSHL	RABPTR	:	4095
	01	FB	000C7		CALLS	#1, SYSSPUT	:	
	04	00	00CE	7\$:	RET		:	4097

; Routine Size: 207 bytes. Routine Base: \$CODE\$ + 21E8

```
4033 4098 1 %sbttl 'build_ini_recs - build system & default records'
4034 4099 1 routine build_ini_recs : novalue =
4035 4100 2 begin
4036 4101 2 ++
4037 4102 2
4038 4103 2
4039 4104 2 FUNCTIONAL DESCRIPTION:
4040 4105 2
4041 4106 2 Build the initial records for the creation of a new UAF file.
4042 4107 2 The user default record is built in the DEFAULT_RECORD
4043 4108 2 buffer and the system manager record is built in RECBUF.
4044 4109 2
4045 4110 2 INPUTS:
4046 4111 2
4047 4112 2 none
4048 4113 2
4049 4114 2 IMPLICIT INPUTS:
4050 4115 2
4051 4116 2 none
4052 4117 2
4053 4118 2 OUTPUTS:
4054 4119 2
4055 4120 2 none
4056 4121 2
4057 4122 2 IMPLICIT OUTPUTS:
4058 4123 2
4059 4124 2 default record is built in DEFAULT_RECORD
4060 4125 2 system record is built in RECBUF
4061 4126 2
4062 4127 2 ROUTINE VALUE:
4063 4128 2
4064 4129 2 none
4065 4130 2
4066 4131 2 SIDE EFFECTS:
4067 4132 2
4068 4133 2 none
4069 4134 2 --
4070 4135 2
4071 4136 2 local
4072 4137 2 user_desc : %bblock [8];
4073 4138 2
4074 4139 2 ch$fill (0, uaf$b_fixed, default_record);
4075 4140 2 default_record[uaf$b_version] = uaf$b_version1;
4076 4141 2 default_record[uaf$b_rtype] = uaf$b_user_id;
4077 4142 2
4078 4143 2 username is blank filled
4079 4144 2
4080 4145 2
4081 4146 2 ch$copy (.defuser<0,8>, defuser+1, %char (' '),
4082 4147 2 uaf$b_username, default_record[uaf$b_username]);
4083 4148 2
4084 4149 2
4085 4150 2 account name is blank filled
4086 4151 2
4087 4152 2
4088 4153 2 ch$copy (.defact<0,8>, defact+1, %char (' '),
4089 4154 2 uaf$b_account, default_record[uaf$b_account]);
```



```
4090 4155 2
4091 4156 2
4092 4157 2 quadword privilege mask
4093 4158 2
4094 4159 2
4095 4160 2 ch$move (8, defpriv, default_record[uaf$q_priv]);
4096 4161 2 ch$move (8, defpriv, default_record[uaf$q_def_priv]);
4097 4162 2
4098 4163 2
4099 4164 2 directory name is counted string
4100 4165 2
4101 4166 2
4102 4167 2 ch$copy (.defdir<0,8> + 1, defdir, %char (' '),
4103 4168 2 uaf$s_defdir, default_record[uaf$t_defdir]);
4104 4169 2
4105 4170 2
4106 4171 2 device name is counted string
4107 4172 2
4108 4173 2
4109 4174 2 ch$copy (.defdev<0,8> + 1, defdev, %char (' '),
4110 4175 2 uaf$s_defdev, default_record[uaf$t_defdev]);
4111 4176 2
4112 4177 2
4113 4178 2 CLI name is counted string
4114 4179 2
4115 4180 2
4116 4181 2 ch$copy (.defcli<0,8> + 1, defcli, %char (' '),
4117 4182 2 uaf$s_defcli, default_record[uaf$t_defcli]);
4118 4183 2
4119 4184 2
4120 4185 2 owner name is counted string
4121 4186 2
4122 4187 2
4123 4188 2 ch$copy (.defowner<0,8> + 1, defowner, %char (' '),
4124 4189 2 uaf$s_owner, default_record[uaf$t_owner]);
4125 4190 2
4126 4191 2
4127 4192 2 login command file name is counted string
4128 4193 2
4129 4194 2
4130 4195 2 ch$copy (.deflgicmd<0,8> + 1, deflgicmd, %char (' '),
4131 4196 2 uaf$s_lgicmd, default_record[uaf$t_lgicmd]);
4132 4197 2
4133 4198 2
4134 4199 2 fill in default CLI tables
4135 4200 2
4136 4201 2 ch$copy (.defclitabl<0,8> + 1, defclitabl, %char (' '),
4137 4202 2 uaf$s_clitables, default_record[uaf$t_clitables]);
4138 4203 2
4139 4204 2 ch$move (8, defpwdlife, default_record[uaf$q_pwd_lifetime]);
4140 4205 2 default_record[uaf$b_pwd_length] = defpwdlength;
4141 4206 2 default_record[uaf$w_grp] = defgrp;
4142 4207 2 default_record[uaf$w_mem] = defmem;
4143 4208 2 default_record[uaf$w_bioltm] = defbioltm;
4144 4209 2 default_record[uaf$l_bytltm] = defbytltm;
4145 4210 2 default_record[uaf$w_diolm] = defdioltm;
4146 4211 2 default_record[uaf$w_fillm] = deffillm;
```

```
4147 4212 2 default_record[uafl_flags] = defflags;
4148 4213 2 default_record[uafl_tqcnt] = deftqcnt;
4149 4214 2 default_record[uafl_prcnt] = defprcnt;
4150 4215 2 default_record[uafl_wsquota] = defwsquota;
4151 4216 2 default_record[uafl_wsextent] = defwsextent;
4152 4217 2 default_record[uafl_dfwsent] = defdfwsent;
4153 4218 2 default_record[uafl_cputim] = defcputim;
4154 4219 2 default_record[uafl_astlm] = defastlm;
4155 4220 2 default_record[uafl_pgflquota] = defpgflquota;
4156 4221 2 default_record[uafl_enqlm] = defenqlm;
4157 4222 2 default_record[uafl_pbytlim] = defpbytlim;
4158 4223 2 default_record[uafl_shrfillm] = defshrfillm;
4159 4224 2 default_record[uafl_pri] = defpri;
4160 4225 2 default_record[uafl_QUEPRI] = defquepri;
4161 4226 2 default_record[uafl_maxjobs] = defmaxjobs;
4162 4227 2 default_record[uafl_maxdetach] = defmaxdetach;
4163 4228 2 default_record[uafl_itquota] = defitquota;
4164 4229 2 default_record[uafl_maxacctjobs] = defmaxacctjobs;
4165 4230 2 default_record[uafl_primedays] = defprimedays;
4166 4231 2 default_record[uafl_network_access_p] = defhours;
4167 4232 2 default_record[uafl_network_access_s] = defhours;
4168 4233 2 default_record[uafl_batch_access_p] = defhours;
4169 4234 2 default_record[uafl_batch_access_s] = defhours;
4170 4235 2 default_record[uafl_local_access_p] = defhours;
4171 4236 2 default_record[uafl_local_access_s] = defhours;
4172 4237 2 default_record[uafl_dialup_access_p] = defhours;
4173 4238 2 default_record[uafl_dialup_access_s] = defhours;
4174 4239 2 default_record[uafl_remote_access_p] = defhours;
4175 4240 2 default_record[uafl_remote_access_s] = defhours;
4176 4241 2
4177 4242 2 ch$fill (0, uafl_fixed, recbuf);
4178 4243 2 recbuf[uafl_version] = uafl_version1;
4179 4244 2 recbuf[uafl_rtype] = uafl_user_id;
4180 4245 2 ch$copy (.sysuser<0,8>, sysuser+1, %char (' '),
4181 4246 2 uafl_username, recbuf[uafl_username]);
4182 4247 2 ch$copy (.sysact<0,8>, sysact+1, %char (' '),
4183 4248 2 uafl_account, recbuf[uafl_account]);
4184 4249 2 ch$move (8, syspriv, recbuf[uafl_priv]);
4185 4250 2 ch$move (8, syspriv, recbuf[uafl_def_priv]);
4186 4251 2 ch$copy (.sysdir<0,8> + 1, sysdir, %char (' '),
4187 4252 2 uafl_defdir, recbuf[uafl_defdir]);
4188 4253 2 ch$copy (.sysdev<0,8> + 1, sysdev, %char (' '),
4189 4254 2 uafl_defdev, recbuf[uafl_defdev]);
4190 4255 2 ch$copy (.syscli<0,8> + 1, syscli, %char (' '),
4191 4256 2 uafl_defcli, recbuf[uafl_defcli]);
4192 4257 2 ch$copy (.sysowner<0,8> + 1, sysowner, %char (' '),
4193 4258 2 uafl_owner, recbuf[uafl_owner]);
4194 4259 2 ch$copy (.syslgicmd<0,8> + 1, deflgicmd, %char (' '),
4195 4260 2 uafl_lgicmd, recbuf[uafl_lgicmd]);
4196 4261 2 ch$copy (.sysclitabl<0,8> + 1, sysclitabl, %char (' '),
4197 4262 2 uafl_clitables, recbuf[uafl_clitables]);
4198 4263 2
4199 4264 2 pddsc[dsc$w_length] = .syspass<0,8>;
4200 4265 2 pddsc[dsc$a_pointer] = syspass+1;
4201 4266 2 user_desc[dsc$w_length] = .sysuser<0,8>;
4202 4267 2 user_desc[dsc$a_pointer] = sysuser+1;
4203 4268 2 $gettim (timadr = time_buf);
```

! Obtain a 16 bit salt

```
4204 2 recbuf[uaf$w_salt] = .time_buf<3*8,16>;
4205 2 recbuf[uaf$b_encrypt] = encrypt;
4206 2 lgishpwd (rec_encrypt_dsc, pwddsc, .recbuf[uaf$b_encrypt],
4207 2 .recbuf[uaf$w_salt], user_desc);
4208 2
4209 2 ch$move (8, syspwdlife, recbuf[uaf$q_pwd_lifetime]);
4210 2 recbuf[uaf$b_pwd_length] = syspwdlength;
4211 2 recbuf[uaf$w_grp] = sysgrp;
4212 2 recbuf[uaf$w_mem] = sysmem;
4213 2 recbuf[uaf$w_bioltm] = sysbioltm;
4214 2 recbuf[uaf$l_bytltm] = sysbytltm;
4215 2 recbuf[uaf$w_diolm] = sysdioltm;
4216 2 recbuf[uaf$w_fillm] = sysfillm;
4217 2 recbuf[uaf$l_flags] = sysflags;
4218 2 recbuf[uaf$w_tqcnt] = systqcnt;
4219 2 recbuf[uaf$w_prcnt] = sysprcnt;
4220 2 recbuf[uaf$l_wsquota] = syswsquota;
4221 2 recbuf[uaf$l_wsextent] = syswsextent;
4222 2 recbuf[uaf$l_dfwscnt] = sysdfwscnt;
4223 2 recbuf[uaf$l_cputim] = syscputim;
4224 2 recbuf[uaf$w_astltm] = sysastltm;
4225 2 recbuf[uaf$l_pgflquota] = syspgflquota;
4226 2 recbuf[uaf$w_enqlm] = sysenqlm;
4227 2 recbuf[uaf$l_pbytltm] = syspbytltm;
4228 2 recbuf[uaf$w_shrfillm] = sysshrfillm;
4229 2 recbuf[uaf$b_pri] = syspri;
4230 2 default_record[uaf$b_QUEPRI] = sysquepri;
4231 2 recbuf[uaf$w_maxdetach] = sysmaxdetach;
4232 2 recbuf[uaf$l_jtquota] = sysjtquota;
4233 2 recbuf[uaf$w_maxjobs] = sysmaxjobs;
4234 2 recbuf[uaf$w_maxdetach] = sysmaxdetach;
4235 2 recbuf[uaf$w_maxacctjobs] = sysmaxacctjobs;
4236 2 recbuf[uaf$b_primedays] = sysprimedays;
4237 2 recbuf[uaf$b_network_access_p] = defhours;
4238 2 recbuf[uaf$b_network_access_s] = defhours;
4239 2 recbuf[uaf$b_batch_access_p] = defhours;
4240 2 recbuf[uaf$b_batch_access_s] = defhours;
4241 2 recbuf[uaf$b_local_access_p] = defhours;
4242 2 recbuf[uaf$b_local_access_s] = defhours;
4243 2 recbuf[uaf$b_dialup_access_p] = defhours;
4244 2 recbuf[uaf$b_dialup_access_s] = defhours;
4245 2 recbuf[uaf$b_remote_access_p] = defhours;
4246 2 recbuf[uaf$b_remote_access_s] = defhours;
4247 1 end;
```

## 03FC 0000 BUILD\_INI RECS:

59	00000000'	00	9E	00002	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	4099
58	00000000'	00	9E	00009	MOVAB	DEFACT+1, R9	
57	00000000'	00	9E	00010	MOVAB	DEFAULT_RECORD, R8	
5E		08	C2	00017	MOVAB	RECBUF, R7	
6E		00	2C	0001A	SUBL2	#8, SP	
		68		00021	MOVCS	#0, (SP), #0, #644, DEFAULT_RECORD	4139



			68	0101	8F	B0	00022	MOVW	#257, DEFAULT_RECORD	4141	
			50	E8	A9	9A	00027	MOVZBL	DEFUSER, R0	4146	
20	20	E9	A9		50	2C	0002B	MOVCS	R0, DEFUSER+1, #32, #32, DEFAULT_RECORD+4	4147	
				04	A8		00031				
20	20		50	FF	A9	9A	00033	MOVZBL	DEFACT, R0	4153	
			69		50	2C	00037	MOVCS	R0, DEFACT+1, #32, #32, DEFAULT_RECORD+52	4154	
				34	A8		0003C				
	019C	C8	10	A9	08	28	0003E	MOVCS	#8, DEFPRIV, DEFAULT_RECORD+412	4160	
	01A4	C8	10	A9	08	28	00045	MOVCS	#8, DEFPRIV, DEFAULT_RECORD+420	4161	
			50		A9	9A	0004C	MOVZBL	DEFDIR, R0	4167	
0040	BF	20	06	A9	50	D6	00050	INCL	R0		
					50	2C	00052	MOVCS	R0, DEFDIR, #32, #64, DEFAULT_RECORD+148	4168	
			50	0094	C8		0005A				
				0D	A9	9A	0005D	MOVZBL	DEFDEV, R0	4174	
20	20	0D	A9		50	D6	00061	INCL	R0		
				74	50	2C	00063	MOVCS	R0, DEFDEV, #32, #32, DEFAULT_RECORD+116	4175	
			50		A8		00069				
20	20		69		69	9A	0006B	MOVZBL	DEFCLI, R0	4181	
					50	D6	0006E	INCL	R0		
				0114	50	2C	00070	MOVCS	R0, DEFCLI, #32, #32, DEFAULT_RECORD+276	4182	
			50	04	C8		00075				
20	20	04	A9		A9	9A	00078	MOVZBL	DEFOWNER, R0	4188	
					50	D6	0007C	INCL	R0		
				54	50	2C	0007E	MOVCS	R0, DEFOWNER, #32, #32, DEFAULT_RECORD+84	4189	
			50		A8		00084				
				05	A9	9A	00086	MOVZBL	DEFLGICMD, R0	4195	
0040	BF	20	05	A9	50	D6	0008A	INCL	R0		
					50	2C	0008C	MOVCS	R0, DEFLGICMD, #32, #64, DEFAULT_RECORD+212	4196	
			50	00D4	C8		00094				
				F5	A9	9A	00097	MOVZBL	DEFCLITABL, R0	4201	
20	20	F5	A9		50	D6	0009B	INCL	R0		
				0134	50	2C	0009D	MOVCS	R0, DEFCLITABL, #32, #32, -	4202	
	0174	C8	18	A9	C8		000A3				
			016A	C8	08	28	000A6	MOVCS	#8, DEFPWDLIFE, DEFAULT_RECORD+372	4204	
			24	A8	06	90	000AD	MOVBL	#6, DEFAULT_RECORD+362	4205	
			0230	C8	00800080	8F	D0	000B2	MOVL	#8388736, DEFAULT_RECORD+36	4207
			020E	C8	1000	8F	3C	000BA	MOVZWL	#4096, DEFAULT_RECORD+560	4209
					00060006	8F	D0	000C1	MOVL	#393222, DEFAULT_RECORD+526	4208
					01D4	C8	D4	000CA	CLRL	DEFAULT_RECORD+468	4212
			020C	C8		02	B0	000CE	MOVW	#2, DEFAULT_RECORD+524	4214
			021C	C8	C8	8F	9A	000D3	MOVZBL	#200, DEFAULT_RECORD+540	4215
			0224	C8	01F4	8F	3C	000D9	MOVZWL	#500, DEFAULT_RECORD+548	4216
			0220	C8	96	8F	9A	000E0	MOVZBL	#150, DEFAULT_RECORD+544	4217
					022C	C8	D4	000E6	CLRL	DEFAULT_RECORD+556	4218
			0212	C8	000A000A	8F	D0	000EA	MOVL	#655370, DEFAULT_RECORD+530	4213
			0228	C8	2710	8F	3C	000F3	MOVZWL	#10000, DEFAULT_RECORD+552	4220
			0216	C8	0014000A	8F	D0	000FA	MOVL	#1310730, DEFAULT_RECORD+534	4221
					0234	C8	D4	00103	CLRL	DEFAULT_RECORD+564	4222
					021A	C8	B4	00107	CLRW	DEFAULT_RECORD+538	4223
			0204	C8	0404	8F	3C	0010B	MOVZWL	#1028, DEFAULT_RECORD+516	4224
			0238	C8	0400	8F	3C	00112	MOVZWL	#1024, DEFAULT_RECORD+568	4228
					0208	C8	D4	00119	CLRL	DEFAULT_RECORD+520	4229
			0202	C8	60	8F	90	00110	MOVBL	#96, DEFAULT_RECORD+514	4230
01DB	C8	18	00		00	F0	00123	INSV	#0, #0, #24, DEFAULT_RECORD+472	4231	
01DB	C8	18	00		00	F0	0012A	INSV	#0, #0, #24, DEFAULT_RECORD+475	4232	
01DE	C8	18	00		00	F0	00131	INSV	#0, #0, #24, DEFAULT_RECORD+478	4233	
01E1	C8	18	00		00	F0	00138	INSV	#0, #0, #24, DEFAULT_RECORD+481	4234	

01E4	C8	18	00	00	FO	0013F	INSV	#0, #0, #24, DEFAULT_RECORD+484	4235
01E7	C8	18	00	00	FO	00146	INSV	#0, #0, #24, DEFAULT_RECORD+487	4236
01EA	C8	18	00	00	FO	0014D	INSV	#0, #0, #24, DEFAULT_RECORD+490	4237
01ED	C8	18	00	00	FO	00154	INSV	#0, #0, #24, DEFAULT_RECORD+493	4238
01F0	C8	18	00	00	FO	0015B	INSV	#0, #0, #24, DEFAULT_RECORD+496	4239
01F3	C8	18	00	00	FO	00162	INSV	#0, #0, #24, DEFAULT_RECORD+499	4240
0284	8F	00	6E	00	2C	00169	MOVCS	#0, (SP), #0, #644, RECBUF	4242
			67	0101	8F	BO	MOVW	#257, RECBUF	4244
			56	20	A9	9A	MOVZBL	SYSUSER, R6	4245
	20	20	21	A9	56	2C	MOVCS	R6, SYSUSER+1, #32, #32, RECBUF+4	4246
				04	A7				
			50	39	A9	9A	MOVZBL	SYSACT, R0	4247
	20	20	3A	A9	50	2C	MOVCS	R0, SYSACT+1, #32, #32, RECBUF+52	4248
				34	A7				
		019C	C7	60	A9	08	MOVCS	#8, SYSPRIV, RECBUF+412	4249
		01A4	C7	60	A9	08	MOVCS	#8, SYSPRIV, RECBUF+420	4250
				50	A9	9A	MOVZBL	SYSDEV, R0	4251
				50	D6	001A0	INCL	R0	
0040	8F	20	54	A9	50	2C	MOVCS	R0, SYSDEV, #32, #64, RECBUF+148	4252
				50	C7	0094			
				50	A9	9A	MOVZBL	SYSCLI, R0	4253
				50	D6	001AD	INCL	R0	
	20	20	5D	A9	50	2C	MOVCS	R0, SYSCLI, #32, #32, RECBUF+116	4254
				74	A7				
				50	A9	9A	MOVZBL	SYSOWNER, R0	4255
				50	D6	001BB	INCL	R0	
	20	20	40	A9	50	2C	MOVCS	R0, SYSOWNER, #32, #32, RECBUF+276	4256
				50	C7	0114			
				50	A9	9A	MOVZBL	SYSCLITABL, R0	4257
				50	D6	001CA	INCL	R0	
	20	20	44	A9	50	2C	MOVCS	R0, SYSCLITABL, #32, #32, RECBUF+84	4258
				54	A7				
				50	A9	9A	MOVZBL	SYSLGICMD, R0	4259
				50	D6	001D6	INCL	R0	
0040	8F	20	05	A9	50	2C	MOVCS	R0, DEFLGICMD, #32, #64, RECBUF+212	4260
				50	C7	00D4			
				50	A9	9A	MOVZBL	SYSCLITABL, R0	4261
				50	D6	001D8	INCL	R0	
	20	20	2F	A9	50	2C	MOVCS	R0, SYSCLITABL, #32, #32, RECBUF+308	4262
					C7	0134			
		0584	C8		A9	9B	MOVZBW	SYSPASS, PWDDSC	4264
		0588	C8		A9	9E	MOVAB	SYSPASS+1, PWDDSC+4	4265
			6E		56	BO	MOVW	R6, USER_DESC	4266
		04	AE		A9	9E	MOVAB	SYSUSER+T, USER_DESC+4	4267
					C7	9F	PUSHAB	TIME BUF	4268
		00000000G	00		01	FB	CALLS	#1, SYSSGETTIM	
		0166	C7		C7	BO	MOVW	TIME BUF+3, RECBUF+358	4269
		0168	C7		02	90	MOVW	#2, RECBUF+360	4270
					5E	DD	PUSHL	SP	4271
			7E		C7	3C	MOVZWL	RECBUF+358, -(SP)	4272
			7E		C7	9A	MOVZBL	RECBUF+360, -(SP)	4271
					C8	9F	PUSHAB	PWDDSC	
					C9	9F	PUSHAB	REC_ENCRYPT_DSC	
		00000000G	00		05	FB	CALLS	#5, LGISHPWD	
0174	C7		68	A9	08	28	MOVCS	#8, SYSPWDLIFE, RECBUF+372	4274
		016A	C7		08	90	MOVW	#8, RECBUF+362	4275



01D8	C7	18	0202	C7	00	00	FO	002C7	INSV	#0, #0, #24, RECBUF+472	4302
01DB	C7	18		00	00	00	FO	002CE	INSV	#0, #0, #24, RECBUF+475	4303
01DE	C7	18		00	00	00	FO	002D5	INSV	#0, #0, #24, RECBUF+478	4304
01E1	C7	18		00	00	00	FO	002DC	INSV	#0, #0, #24, RECBUF+481	4305
01E4	C7	18		00	00	00	FO	002E3	INSV	#0, #0, #24, RECBUF+484	4306
01E7	C7	18		00	00	00	FO	002EA	INSV	#0, #0, #24, RECBUF+487	4307
01EA	C7	18		00	00	00	FO	002F1	INSV	#0, #0, #24, RECBUF+490	4308
01ED	C7	18		00	00	00	FO	002F8	INSV	#0, #0, #24, RECBUF+493	4309
01F0	C7	18		00	00	00	FO	002FF	INSV	#0, #0, #24, RECBUF+496	4310
01F3	C7	18		00	00	00	FO	00306	INSV	#0, #0, #24, RECBUF+499	4311
						04	00	0030D	RET		4312

  

24	A7	00010004	8F	D0	0024A	MOVL	#65540, RECBUF+36	4277
0230	C7	5000	8F	3C	00252	MOVZWL	#20480, RECBUF+560	4279
020E	C7	000C000C	8F	D0	00259	MOVL	#78644, RECBUF+526	4278
		01D4	C7	D4	00262	CLRL	RECBUF+468	4282
0212	C7	00140014	8F	D0	00266	MOVL	#1310740, RECBUF+530	4283
020A	C7	000A0000	8F	D0	0026F	MOVL	#655360, RECBUF+522	4296
021C	C7	015E	8F	3C	00278	MOVZWL	#350, RECBUF+540	4285
0224	C7	0400	8F	3C	0027F	MOVZWL	#1024, RECBUF+548	4286
0220	C7	96	8F	9A	00286	MOVZBL	#150, RECBUF+544	4287
		022C	C7	D4	0028C	CLRL	RECBUF+556	4288
0228	C7	2710	8F	3C	00290	MOVZWL	#10000, RECBUF+552	4290
0216	C7	00140014	8F	D0	00297	MOVL	#1310740, RECBUF+534	4291
		0234	C7	D4	002A0	CLRL	RECBUF+564	4292
		021A	C7	B4	002A4	CLRW	RECBUF+538	4293
0204	C7		04	90	002AB	MOVB	#4, RECBUF+516	4294
0205	C8		04	90	002AD	MOVB	#4, DEFAULT RECORD+517	4295
0238	C7	0400	8F	3C	002B2	MOVZWL	#1024, RECBUF+568	4297
		020A	C7	B4	002B9	CLRW	RECBUF+522	4299
		0206	C7	D4	002BD	CLRL	RECBUF+518	4298
		60	8F	90	002C1	MOVB	#96, RECBUF+514	4301

; Routine Size: 782 bytes. Routine Base: \$CODE\$ + 22B7

```
4249 4313 1 %sbttl 'get_user_record - get username and lookup record'
4250 4314 1 routine get_user_record (lock_record, permanent_ok) =
4251 4315 1 begin
4252 4316 1
4253 4317 1 ++
4254 4318 1
4255 4319 1 FUNCTIONAL DESCRIPTION:
4256 4320 1
4257 4321 1 This routine pulls the next token out of the command
4258 4322 1 buffer, assuming it is the username, and looks up the
4259 4323 1 UAF record for that name. If the record is found,
4260 4324 1 it is loaded into RECBUF (by routine LOCATE_USER).
4261 4325 1
4262 4326 1 INPUTS:
4263 4327 1
4264 4328 1 LOCK_RECORD - specifies that the GET shall lock the record
4265 4329 1 PERMANENT_OK - specifies that the DEFAULT and SYSTEM records are allowed
4266 4330 1
4267 4331 1 IMPLICIT INPUTS:
4268 4332 1
4269 4333 1 TOKENPTR - address of delimiter following last token processed,
4270 4334 1 which was the command name.
4271 4335 1 TOKENLEN - global variable to contain length of current token
4272 4336 1
4273 4337 1 OUTPUTS:
4274 4338 1
4275 4339 1 none
4276 4340 1
4277 4341 1 IMPLICIT OUTPUTS:
4278 4342 1
4279 4343 1 none
4280 4344 1
4281 4345 1 ROUTINE VALUE:
4282 4346 1
4283 4347 1 true -> user record found
4284 4348 1 false -> user record not found
4285 4349 1
4286 4350 1 SIDE EFFECTS:
4287 4351 1
4288 4352 1 none
4289 4353 1 --
4290 4354 1
4291 4355 1 builtin
4292 4356 1 nullparameter;
4293 4357 1
4294 4358 1
4295 4359 1 Is this the second phase of a RENAME?
4296 4360 1
4297 4361 1 if .rename_ph2
4298 4362 1 then
4299 4363 1 begin
4300 4364 1 if .netuaf exists
4301 4365 1 then adjust_proxy (update_records);
4302 4366 1 ch$move (.olduserlen, oldusername, .tokenptr);
4303 4367 1 tokenlen = .olduserlen;
4304 4368 1 rename_ph2 = false;
4305 4369 1 end
```

```
4306 4370
4307 4371
4308 4372 Not the second phase of a RENAME. Get first token, and if this
4309 4373 is the first phase of a RENAME (COPY phase), save token for
4310 4374 the second phase (REMOVE phase).
4311 4375
4312 4376 else
4313 4377     begin
4314 4378         Get token
4315 4379
4316 4380         if not cli$present (sd_token1)
4317 4381             or not cli$get_value (sd_token1, tokendsc)
4318 4382             or .tokenlen eql 0
4319 4383             then return LIB$SIGNAL(UAF$_NOUSERNAME);
4320 4384
4321 4385         If the third argument is present, this is the first phase of a
4322 4386         RENAME, so save the token for the next call
4323 4387
4324 4388         if not nullparameter (3)
4325 4389             then
4326 4390                 begin
4327 4391                     ch$copy (.tokenlen, .tokenptr, ' ', uaf$_username, oldusername);
4328 4392                     olduserlen = .tokenlen;
4329 4393                     rename_ph2 = true;
4330 4394                     end;
4331 4395                 end;
4332 4396
4333 4397         if not .permanent_ok
4334 4398             then
4335 4399                 begin
4336 4400                     if ch$seq (.defuser<0,8>, defuser+1, .tokenlen, .tokenptr, ' ')
4337 4401                         then
4338 4402                             if nullparameter (3)
4339 4403                                 then
4340 4404                                     return LIB$SIGNAL(UAF$_REMDEF)
4341 4405                                 else
4342 4406                                     return LIB$SIGNAL(UAF$_RENDEF);
4343 4407
4344 4408                     if ch$seq (.sysuser<0,8>, sysuser+1, .tokenlen, .tokenptr, ' ')
4345 4409                         then
4346 4410                             begin
4347 4411                                 if nullparameter (3)
4348 4412                                     then
4349 4413                                         return LIB$SIGNAL(UAF$_REMSYS)
4350 4414                                     else
4351 4415                                         return LIB$SIGNAL(UAF$_RENSYS);
4352 4416                             end;
4353 4417                         end;
4354 4418
4355 4419         if locate_user (.tokenlen, .tokenptr, .lock_record)
4356 4420             then return true;
4357 4421
4358 4422         if .rmserr eql rms$_rnf
4359 4423             then
4360 4424                 LIB$SIGNAL(UAF$_BADUSR, 2, .tokenlen, .tokenptr)
4361 4425             else
4362 4426                 LIB$SIGNAL(UAF$_GETERR, 0, .rmserr);
```

```
07FC 00000 GET_USER_RECORD:
      5A 00000000G 00 9E 00002  .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10      4314
      59 00000000 00 9E 00009  MOVAB      LIB$SIGNAL, R10
      58 00000000 00 9E 00010  MOVAB      SD TOKEN1, R9
      57 00000000 00 9E 00017  MOVAB      RENAME_PH2, R8
      1F 00000000 68 E9 0001E  MOVAB      TOKENLEN, R7
      07      10  A7 E9 00021  BLBC      RENAME_PH2, 2$      4361
      E796  CF      01 FB 00027  BLBC      NETUAF_EXISTS, 1$      4364
      56      04  A8 D0 0002C  CLRL      -(SP)      4365
      50      04  A7 D0 00030  CALLS     #1, ADJUST_PROXY
      60      08  56 28 00034  MOVBL     OLDUSERLEN, R6      4366
      67      56 B0 00039  MOVBL     TOKENPTR, R0
      68 94 0003C  MOVCL     R6, OLDUSERNAME, (R0)
      45 11 0003E  MOVW      R6, TOKENLEN      4367
      59 DD 00040  CLRB      RENAME_PH2      4368
      00      01 FB 00042  BRB       5$      4361
      12      50 E9 00049  BRB       5$      4361
      57 DD 0004C  PUSHL     R9      4380
      59 DD 0004E  CALLS     #1, CLIS$PRESENT
      00      02 FB 00050  BLBC      R0, 3$      4381
      04      50 E9 00057  PUSHL     R9
      00000000G 00      67 B5 0005A  CALLS     #2, CLIS$GET_VALUE
      04      08 12 0005C  BLBC      R0, 3$
      00000000G 00      08 12 0005C  TSTW      TOKENLEN      4382
      03      6C 91 00066  BNEQ      4$      4383
      0C      1A 1F 00069  PUSHL     #UAF$_NOUSERNAME
      56      04  A7 D0 00073  BRB       11$      4388
      60      08  56 2C 00077  CMPB      (AP), #3
      04  A8 56 D0 0007E  BLSSU     5$      4391
      68      01 90 00082  TSTL      12(AP)
      5C      08 AC E8 00085  BEQL      5$
      51 0118 C9 9A 00089  MOVZWL    TOKENLEN, R6
      50      04  A7 D0 0008E  MOVL      TOKENPTR, R0
      67      20 0119 C9 51 2D 00092  MOVCL     R6, (R0), #32, #32, OLDUSERNAME
      60      1A 12 0009A  MOVBL     R6, OLDUSERLEN      4392
      03      6C 91 0009C  MOVBL     #1, RENAME_PH2      4393
      0C      05 1F 0009F  BLBS      PERMANENT_OK, 12$      4397
      00000000G 8F DD 000A6  MOVZBL    DEFUSER, R1      4400
      00000000G 33 11 000AC  MOVL      TOKENPTR, R0
      8F DD 000AE  CMPC5     R1, DEFUSER+1, #32, TOKENLEN, (R0)
      03      6C 91 0009C  BNEQ      8$      4402
      0C      05 1F 0009F  CMPB      (AP), #3
      00000000G 8F DD 000A6  BLSSU     6$      4404
      00000000G 33 11 000AC  TSTL      12(AP)
      8F DD 000AE  BNEQ      7$      4406
      03      6C 91 0009C  PUSHL     #UAF$_RENDEF
      0C      05 1F 0009F  BRB       11$
      00000000G 8F DD 000AE  PUSHL     #UAF$_RENDEF
```

67	20	0151	51	0150	2B	11	000B4	BRB	11\$		
			50	04	C9	9A	000B6	MOVZBL	SYSUSER, R1		4408
			C9		A7	D0	000BB	MOVL	TOKENPTR, R0		
					51	2D	000BF	CMPC5	R1, SYSUSER+1, #32, TOKENLEN, (R0)		
					60		000C6				
			03		1C	12	000C7	BNEQ	12\$		
					6C	91	000C9	CMPB	(AP), #3		4411
					05	1F	000CC	BLSSU	9\$		
				0C	AC	D5	000CE	TSTL	12(AP)		
					08	12	000D1	BNEQ	10\$		
				00000000G	8F	DD	000D3	PUSHL	#UAF\$_REMSYS		4413
					06	11	000D9	BRB	11\$		
				00000000G	8F	DD	000DB	PUSHL	#UAF\$_REMSYS		4415
			6A		01	FB	000E1	CALLS	#1, LIB\$SIGNAL		
						04	000E4	RET			
				04	AC	DD	000E5	PUSHL	LOCK RECORD		4419
				04	A7	DD	000E8	PUSHL	TOKENPTR		
			7E		67	3C	000EB	MOVZWL	TOKENLEN, -(SP)		
			00000000V		03	FB	000EE	CALLS	#3, LOCATE_USER		
					50	E9	000F5	BLBC	R0, 13\$		
					50	01	000F8	MOVL	#1, R0		4420
						04	000F8	RET			
			50	18	A7	D0	000FC	MOVZBL	RMSERR, R0		4422
			000182B2		50	D1	00100	CMPL	R0, #98994		
					13	12	00107	BNEQ	14\$		
				04	A7	DD	00109	PUSHL	TOKENPTR		4424
			7E		67	3C	0010C	MOVZWL	TOKENLEN, -(SP)		
					02	DD	0010F	PUSHL	#2		
				00000000G	8F	DD	00111	PUSHL	#UAF\$_BADUSR		
			6A		04	FB	00117	CALLS	#4, LIB\$SIGNAL		
					0D	11	0011A	BRB	15\$		
					50	DD	0011C	PUSHL	R0		4426
					7E	D4	0011E	CLRL	-(SP)		
				00000000G	8F	DD	00120	PUSHL	#UAF\$_GETERR		
			6A		03	FB	00126	CALLS	#3, LIB\$SIGNAL		
					50	D4	00129	CLRL	R0		4428
						04	0012B	RET			

; Routine Size: 300 bytes. Routine Base: \$CODE\$ + 25C5



locate\_user - lookup user record in UAF

```
4366 4429 1 %sbttl 'locate_user - lookup user record in UAF'
4367 4430 1 routine locate_user (size, buffer, lock_record) =
4368 4431 2 begin
4369 4432 3
4370 4433 4 ++
4371 4434 5
4372 4435 6 FUNCTIONAL DESCRIPTION:
4373 4436 7
4374 4437 8     Routine to locate a user record in the UAF file.
4375 4438 9
4376 4439 10 INPUTS:
4377 4440 11
4378 4441 12     SIZE - size of the username string
4379 4442 13     BUFFER - address of the username string
4380 4443 14     LOCK_RECORD - specifies that the GET shall lock the record
4381 4444 15
4382 4445 16 IMPLICIT INPUTS:
4383 4446 17
4384 4447 18     UAFRAB - RMS data structure for SYSUAF.DAT
4385 4448 19
4386 4449 20 OUTPUTS:
4387 4450 21
4388 4451 22     none
4389 4452 23
4390 4453 24 IMPLICIT OUTPUTS:
4391 4454 25
4392 4455 26     If record is found, RECBUF contains the located record.
4393 4456 27
4394 4457 28 ROUTINE VALUE:
4395 4458 29
4396 4459 30     true -> record found
4397 4460 31     false -> record not found
4398 4461 32
4399 4462 33 SIDE EFFECTS:
4400 4463 34
4401 4464 35     none
4402 4465 36
4403 4466 37 --
4404 4467 38 local
4405 4468 39     found;
4406 4469 40
4407 4470 41 found = true;
4408 4471 42
4409 4472 43 ch$copy (.size, .buffer, %char (' '),
4410 4473 44     uaf$s_username, recbuf[uaf$t_username]);
4411 4474 45
4412 4475 46 if .lock_record
4413 4476 47 then uaf$rab[ra$b$l_rop] = ra$b$m_rlk
4414 4477 48 else uaf$rab[ra$b$l_rop] = ra$b$m_rrl or ra$b$m_nlk;
4415 4478 49
4416 4479 50 if not (rmserr = get_uaf_record ())
4417 4480 51 then found = false;
4418 4481 52
4419 4482 53 return .found;
4420 4483 54 end;
```

locate\_user - lookup user record in UAF

16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 B11ss-32 V4.0-742 Page 163  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32:1 (33)

UAI  
VO4

				00FC	00000	LOCATE_USER:				
						.WORD	Save R2,R3,R4,R5,R6,R7	:	4430	
		57	00000000'	00	9E	00002	MOVAB	UAFRAB+4, R7	:	
		56		01	D0	00009	MOVL	#1, FOUND	:	4470
20	20	08	BC	AC	2C	0000C	MOVCS	SIZE, @BUFFER, #32, #32, RECBUF+4	:	4473
			FA18	C7		00013			:	
		09		AC	E9	00016	BLBC	LOCK RECORD, 1\$	:	4475
		67	00080000	BF	D0	0001A	MOVL	#524288, UAFRAB+4	:	4476
				07	11	00021	BRB	2\$	:	
		67	00100008	BF	D0	00023	1\$: MOVL	#1048584, UAFRAB+4	:	4477
	00000000V	00		00	FB	0002A	2\$: CALLS	#0, GET_UAF_RECORD	:	4479
	FA0C	C7		50	D0	00031	MOVL	R0, RMSERR	:	
		02		50	E8	00036	BLBS	R0, 3\$	:	
				56	D4	00039	CLRL	FOUND	:	4480
		50		56	D0	0003B	3\$: MOVL	FOUND, R0	:	4482
					04	0003E	RET		:	4483

; Routine Size: 63 bytes, Routine Base: \$CODES + 26F1

```
get_uaf_record - routine to deal with record lo
4484 1 %sbttl 'get_uaf_record - routine to deal with record locking'
4485 routine get_uaf_record =
4486 begin
4487
4488 ++
4489
4490 FUNCTIONAL DESCRIPTION:
4491
4492     A common routine to GET records from SYSUAF.DAT, deal with retries
4493     when the record is locked.
4494
4495 INPUTS:
4496
4497     none
4498
4499 IMPLICIT INPUTS:
4500
4501     UAFRAB - RMS data structure for SYSUAF.DAT
4502
4503 OUTPUTS:
4504
4505     none
4506
4507 IMPLICIT OUTPUTS:
4508
4509     RECBUF - The user's record
4510
4511 ROUTINE VALUE:
4512
4513     RMS status code
4514
4515 SIDE EFFECTS:
4516
4517     none
4518
4519 --
4520 local
4521     counter,
4522     success;
4523
4524     ! number of retries remaining
4525
4526 If anybody's record is locked it shouldn't remain that way for long.
4527 Also, ignore the special system password record.
4528
4529 counter = retry_rlk;
4530 while ( ((success = $get (rab = uafrab)) eql rms$ rlk)
4531         or ch$eq1(UAF$S USERNAME, RECBUF[UAF$T USERNAME],
4532                 17, uplit(ascii('<System+Password>'), %c' ')) )
4533         and ( (counter = .counter - 1) geq 0)
4534 do
4535     if $schdwk (daytim = wakedelta) then $hiber;
4536
4537 .success
4538 end;
```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 00878 P.AFG: .ASCII \<System+Password>\<0><0><0>
00 00 00 3E 64 00887
:

.PSECT $CODE$,NOWRT,2
003C 00000 GET_UAF_RECORD:
      54 00000000' 08 D0 00002 .WORD Save R2,R3,R4,R5 : 4485
      00 00000000' 00 9F 00005 1$: MOVL #8, COUNTER : 4529
00000000G 00 01 FB 0000B PUSHAB UAFRAB : 4530
      55 50 D0 00012 CALLS #1, SYSSGET
000182AA 8F 55 D1 00015 MOVL R0, SUCCESS
      10 13 0001C CMPL SUCCESS, #98986
11 20 00000000' 00 20 2D 0001E BEQL 2$
      00 00000000' 00 00 00027 CMPCS #32, RECBUF+4, #32, #17, P.AFG : 4531
      21 12 0002C BNEQ 3$
      54 D7 0002E 2$: DECL COUNTER : 4533
      1D 19 00030 BLSS 3$
      7E D4 00032 CLRL -(SP) : 4535
      00 9F 00034 PUSHAB WAKDELTA
      7E 7C 0003A CLRQ -(SP)
00000000G 00 04 FB 0003C CALLS #4, SYSSSCHDWK
      BF 50 E9 00043 BLBC R0, 1$
00000000G 00 00 FB 00046 CALLS #0, SYSSHIBER
      86 11 0004D BRB 1$
      50 55 D0 0004F 3$: MOVL SUCCESS, R0 : 4538
      04 00052 RET

```

; Routine Size: 83 bytes, Routine Base: \$CODE\$ + 2730

get\_cmd\_line - input user command line

```
4478 4539 1 %sbttl 'get_cmd_line - input user command line'
4479 4540 1 routine get_cmd_line =
4480 4541 1 begin
4481 4542 1
4482 4543 1 ++
4483 4544 1
4484 4545 1 FUNCTIONAL DESCRIPTION:
4485 4546 1
4486 4547 1 This routine reads in the command line from the user,
4487 4548 1 reading additional lines if continuation is specified by
4488 4549 1 a '-' as the last character on the input line.
4489 4550 1 A zero byte is inserted following the last input character read.
4490 4551 1
4491 4552 1 INPUTS:
4492 4553 1
4493 4554 1 none
4494 4555 1
4495 4556 1 IMPLICIT INPUTS:
4496 4557 1
4497 4558 1 CMDBUF - buffer to receive the user's command line
4498 4559 1 CMDBUFLEN - literal length of CMDBUF
4499 4560 1
4500 4561 1 OUTPUTS:
4501 4562 1
4502 4563 1 none
4503 4564 1
4504 4565 1 IMPLICIT OUTPUTS:
4505 4566 1
4506 4567 1 CMDBUF is filled with command line
4507 4568 1
4508 4569 1 ROUTINE VALUE:
4509 4570 1
4510 4571 1 none
4511 4572 1
4512 4573 1 SIDE EFFECTS:
4513 4574 1
4514 4575 1 none
4515 4576 1 --
4516 4577 1
4517 4578 1 map
4518 4579 1 cmdlindsc: vector;
4519 4580 1
4520 4581 1 local
4521 4582 1 ptr
4522 4583 1 buflen;
4523 4584 1
4524 4585 1
4525 4586 1 Prompt with normal prompt string until some input is supplied.
4526 4587 1
4527 4588 1
4528 4589 1 do
4529 4590 1 ask (accprmt, cmdbuf[0], cmdbuflen)
4530 4591 1 until
4531 4592 1 .insize neq 0;
4532 4593 1
4533 4594 1
4534 4595 1 Now that line has been read, continue reading until
```



get\_cmd\_line - input user command line

```
4535      | last character is not a '-'.
4536      |
4537      |
4538      ptr = .insize - 1;          ! index to last character read
4539      buflen = cmdbuflen;       ! initial buffer size
4540
4541      while
4542      .cndbuf[.ptr] eq1 '-'
4543      do
4544      begin
4545          |
4546          | Read starting at the position of the '-'. Then adjust the
4547          | index and remaining length to point to the last character
4548          | of the next input.
4549          |
4550          |
4551          buflen = .buflen - (.insize + 1);
4552          ask (accprmt2, cndbuf[.ptr], .buflen);
4553          if (.ptr + .insize) lss cndbufmax
4554          then
4555              ptr = .ptr + .insize - 1
4556          else
4557              begin
4558                  LIB$SIGNAL(UAF$_CMDTOOLONG, 1, cndbufmax);
4559                  return false;
4560              end;
4561          end;
4562
4563      cmdlindsc [0] = .ptr + 1;
4564      cmdlindsc [1] = cndbuf;
4565
4566      return true;
4567
4568      1 end;
```

			003C 00000 GET_CMD_LINE:			
	55	00000000V	00 9E 00002	WORD	Save R2,R3,R4,R5	4540
	54	00000000'	00 9E 00009	MOVAB	ASK, R5	
	7E	0400	8F 3C 00010	MOVZWL	CMDBUF, R4	4590
			54 DD 00015	1\$:	#1024, -(SP)	
		00000000'	00 9F 00017	PUSHL	R4	
	65		03 FB 0001D	PUSHAB	ACCPMPT	
	50	0990	C4 D0 00020	CALLS	#3, ASK	4592
			E9 13 00025	MOVL	INSIZE, R0	
	52	FF	A0 9E 00027	BEQL	1\$	4599
	53	0400	8F 3C 0002B	MOVAB	-1(R0), PTR	4600
51	52		54 C1 00030	MOVZWL	#1024, BUFLN	4603
	2D		61 91 00034	ADDL3	R4, PTR, R1	
			40 12 00037	CMPB	(R1), #45	
				BNEQ	4\$	
50	53	0990	C4 C3 00039	SUBL3	INSIZE, BUFLN, R0	4613

get\_cmd\_line - input user command line

G 7  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1Page 168  
(35)

	53	FF	A0	9E	0003F	MOVAB	-1(R0), BUFLN		
			0A	BB	00043	PUSHR	#M<R1,R3>		4614
		00000000'	00	9F	00045	PUSHAB	ACPRMPT2		
	65		03	FB	00048	CALLS	#3, ASK		
50	52	0990	C4	C1	0004E	ADDL3	INSIZE, PTR, R0		4615
	8F		50	D1	00054	CMPL	R0, #508		
			06	18	00058	BGEQ	3\$		
	52	FF	A0	9E	0005D	MOVAB	-1(R0), PTR		4617
			CD	11	00061	BRB	2\$		
	7E	01FC	8F	3C	00063	MOVZWL	#508, -(SP)		4620
			01	DD	00068	PUSHL	#1		
		00000000'	8F	DD	0006A	PUSHL	#UAF\$ CMDTOOLONG		
	00		03	FB	00070	CALLS	#3, LIB\$SIGNAL		
			13	11	00077	BRB	5\$		4621
	00000000'	00	A2	9E	00079	MOVAB	1(R2), CMDLINDSC		4626
	00000000'	00	64	9E	00081	MOVAB	CMDBUF, CMDLINDSC+4		4627
			01	D0	00088	MOVL	#1, R0		4629
	50			04	0008B	RET			
			50	D4	0008C	CLRL	R0		4631
				04	0008E	RET			

; Routine Size: 143 bytes, Routine Base: \$CODE\$ + 2783

ask - prompt terminal for input

```
4572 4632 1 %sbttl 'ask - prompt terminal for input'
4573 4633 1 global routine ask (string, buffer, len) : novalue =
4574 4634 1 begin
4575 4635 1
4576 4636 1 ++
4577 4637 1
4578 4638 1 FUNCTIONAL DESCRIPTION:
4579 4639 1
4580 4640 1 Routine to prompt user for input. The input string is read
4581 4641 1 into the user specified buffer and the size read
4582 4642 1 is placed in the global INSIZE.
4583 4643 1
4584 4644 1 INPUTS:
4585 4645 1
4586 4646 1 STRING - the address of a counted ascii prompt string
4587 4647 1 BUFFER - address of the input buffer
4588 4648 1 LEN - length of the input buffer
4589 4649 1
4590 4650 1 IMPLICIT INPUTS:
4591 4651 1
4592 4652 1 none
4593 4653 1
4594 4654 1 OUTPUTS:
4595 4655 1
4596 4656 1 none
4597 4657 1
4598 4658 1 IMPLICIT OUTPUTS:
4599 4659 1
4600 4660 1 INSIZE - size of the input string
4601 4661 1
4602 4662 1 ROUTINE VALUE:
4603 4663 1
4604 4664 1 none
4605 4665 1
4606 4666 1 SIDE EFFECTS:
4607 4667 1
4608 4668 1 none
4609 4669 1 --
4610 4670 1
4611 4671 1 map
4612 4672 1 buffer : ref vector[.byte];
4613 4673 1
4614 4674 1 inrab[rab$l_pbf] = .string + 1; ! prompt string address
4615 4675 1 inrab[rab$b_psz] = . (.string)<0,8>; ! prompt size
4616 4676 1 inrab[rab$l_ubf] = .buffer; ! buffer address
4617 4677 1 inrab[rab$w_usz] = .len; ! buffer size
4618 4678 1
4619 4679 1
4620 4680 1 If end of file encountered on get (either ^Z from terminal or
4621 4681 1 end of file on indirect command file) then take exit path.
4622 4682 1
4623 4683 1
4624 4684 1 if $get (rab=inrab) eql rms$_eof
4625 4685 1 then exit_uaf ();
4626 4686 1
4627 4687 1 if (insize = .inrab[rab$w_rsz]) neq 0 ! get input size
4628 4688 1 then
```

ask - prompt terminal for input

```
4629      begin
4630      incru i to .insize - 1
4631      do
4632      begin
4633      if .buffer[i] gequ 'a' and .buffer[i] lequ 'z'
4634      then buffer[i] = .buffer[i] and not %o'040';
4635      end;
4636      end;
4637      end;
```

! Upcasing is done here because the CVT  
! option does not work under batch.

0090	C2	04	52	00000000'	00	0004	00000	.ENTRY	ASK, Save R2	4633
		0094	AC		01	9E	00002	MOVAB	INSIZE, R2	
		0084	C2	04	01	C1	00009	ADDL3	#1, STRING, INRAB+48	4674
		0080	C2	08	BC	90	00010	MOVB	@STRING, INRAB+52	4675
				0C	AC	D0	00016	MOVL	BUFFER, INRAB+36	4676
				60	AC	B0	0001C	MOVW	LEN, INRAB+32	4677
		00000000G	00		A2	9F	00022	PUSHAB	INRAB	4684
		0001827A	8F		01	FB	00025	CALLS	#1, SYS\$GET	
					50	D1	0002C	CMPL	R0, #98938	
		00000000V	00		07	12	00033	BNEQ	1\$	
			62	0082	00	FB	00035	CALLS	#0, EXIT_UAF	4685
					C2	3C	0003C	MOVZWL	INRAB+34, INSIZE	4687
					24	13	00041	BEQL	5\$	
51			62		01	C3	00043	SUBL3	#1, INSIZE, R1	4690
					50	D4	00047	CLRL	1	
		61	8F	08	17	11	00049	BRB	4\$	
					BC	40	91	0004B	2\$:	
		7A	8F	08	0D	1F	00051	CMPB	@BUFFER[1], #97	4693
					BC	40	91	00053	BLSSU	3\$
		08	BC40		05	1A	00059	CMPB	@BUFFER[1], #122	
					20	8A	0005B	BGTRU	3\$	
			51		50	D6	00060	BICB2	#32, @BUFFER[1]	4694
					50	D1	00062	INCL	1	4690
					E4	1B	00065	CMPL	1, R1	
					04	00067	5\$:	BLEQU	2\$	
								RET		4697

; Routine Size: 104 bytes, Routine Base: \$CODE\$ + 2812

```
4639 4698 1 %sbttl 'fmt_sys_msg - output system message file message'
4640 4699 1 global routine fmt_sys_msg (faostr, msgid, p1) : novalue =
4641 4700 begin
4642 4701
4643 4702 ++
4644 4703
4645 4704 FUNCTIONAL DESCRIPTION:
4646 4705
4647 4706 This routine outputs an error message followed by
4648 4707 the text found in the system message file for the
4649 4708 error condition. If the message is not found, the message
4650 4709 ID itself is printed.
4651 4710
4652 4711 INPUTS:
4653 4712
4654 4713 FAOSTR - address of counted ascii message to be printed
4655 4714 MSGID - error number
4656 4715 P1 - the first of possibly several parameters to FAO
4657 4716
4658 4717 IMPLICIT INPUTS:
4659 4718
4660 4719 None
4661 4720
4662 4721 OUTPUTS:
4663 4722
4664 4723 None
4665 4724
4666 4725 IMPLICIT OUTPUTS:
4667 4726
4668 4727 None
4669 4728
4670 4729 ROUTINE VALUE:
4671 4730
4672 4731 None
4673 4732
4674 4733 SIDE EFFECTS:
4675 4734
4676 4735 None
4677 4736 --
4678 4737
4679 4738 local
4680 4739 buffer : vector [200, byte], ! buffer to receive message
4681 4740 bufdsc : vector [2, long], ! string descriptor
4682 4741 code; ! save return code
4683 4742
4684 4743 bufdsc[0] = 200; ! construct string descriptor
4685 4744 bufdsc[1] = buffer;
4686 4745
4687 4746 code = $getmsg (msgid = .msgid, msglen = bufdsc[0], bufadr = bufdsc[0]);
4688 4747
4689 4748
4690 4749 ! Output internal message. Then output system error or error number.
4691 4750
4692 4751 faodsc[dsc$w_length] = . (.faostr)<0,8>; ! input string descriptor
4693 4752 faodsc[dsc$a_pointer] = .faostr+1;
4694 4753 P $faol (ctrstr = faodsc
4695 4754 P outlen = outrab[rab$w_rs2],
```



```

4696      P 4755      2      outbuf = disdsc,
4697      4756      prmlst = p1);
4698      4757
4699      4758      $put (rab = outrab);
4700      4759
4701      4760      if .code egl ss$ msgnotfnd
4702      4761      then LIB$SIGNAL(DAF$_SYMSMSG2, 1, .msgid)
4703      4762      else LIB$SIGNAL(UAF$_SYMSMSG1, 1, bufdsc[0]);
4704      4763      1      end;

```

.EXTRN SYSSGETMSG, SYSSFAOL

```

.ENTRY    FMT SYS-MSG, Save R2,R3
MOVAB    FAODSC,-R3
MOVAB    -204(SP), SP
MOVZBL   #200, BUFDC
MOVAB    BUFFER, BUFDC+4
MOVQ     #15, -(SP)
PUSHAB   BUFDC
PUSHAB   BUFDC
PUSHL    MSGID
CALLS    #5, SYS$GETMSG
MOVL     R0, CODE
MOVZBW   @FAOSTR, FAODSC
ADDL3    #1, FAOSTR, FAODSC+4
PUSHAB   P1
PUSHAB   DISDC
PUSHAB   OUTRAB+34
PUSHL    R3
CALLS    #4, SYS$FAOL
PUSHAB   OUTRAB
CALLS    #1, SYS$PUT
CMLP     CODE, #1569
BNEQ     1$
PUSHL    MSGID
PUSHL    #1
PUSHL    #UAF$_SYSMSG2
BRB      2$
PUSHL    SP
PUSHL    #1
PUSHL    #UAF$ SYSMSG1
CALLS    #3, LIB$SIGNAL
RET

```

4699  
4743  
4744  
4746  
4751  
4752  
4756  
4758  
4760  
4761  
4762  
4763

; Routine Size: 125 bytes, Routine Base: \$CODE\$ + 287A

faout - output formatted message

```
4706 4764 1 %sbttl 'faout - output formatted message'
4707 4765 1 global routine faout (string, p1) =
4708 4766 begin
4709 4767
4710 4768 ++
4711 4769
4712 4770 FUNCTIONAL DESCRIPTION:
4713 4771
4714 4772 Routine to output a formatted string.
4715 4773
4716 4774 INPUTS:
4717 4775
4718 4776 STRING - address of a counted ASCII FAO control string.
4719 4777 P1 - the first of possibly several parameters to FAO
4720 4778
4721 4779 IMPLICIT INPUTS:
4722 4780
4723 4781 none
4724 4782
4725 4783 OUTPUTS:
4726 4784
4727 4785 none
4728 4786
4729 4787 IMPLICIT OUTPUTS:
4730 4788
4731 4789 none
4732 4790
4733 4791 ROUTINE VALUE:
4734 4792
4735 4793 FAOUT always returns FALSE, as it is often used on the
4736 4794 return from an error condition.
4737 4795
4738 4796 SIDE EFFECTS:
4739 4797
4740 4798 none
4741 4799 --
4742 4800
4743 4801
4744 4802 faodsc[dsc$w_length] = . (.string)<0,8>; ! input string descriptor
4745 4803 faodsc[dsc$a_pointer] = .string+1;
4746 4804 $faol (ctrstr = faodsc,
4747 4805 outlen = outrab[rab$w_rsz],
4748 4806 outbuf = disdsc,
4749 4807 prmlst = p1);
4750 4808
4751 4809 $put (rab = outrab);
4752 4810
4753 4811 false
4754 4812 end;
```

```
52 00000000' 00 0004 00000
62 04 BC 9E 00002
BC 9B 00009
```

```
.ENTRY FAOUT, Save R2
MOVAB FAODSC, R2
MOVZBW @STRING, FAODSC
```

```
: 4765
:
: 4802
```

**faout - output formatted message**

M 7  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742 Page 174  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 (38)

04 A2 04 AC

08  
F8  
072E

000000000G 00

000000000G 00

070c

01	C1	0000D
AC	9F	00013
A2	9F	00016
C2	9F	00019
52	DD	0001D
04	FB	0001F
C2	9F	00026
01	FB	0002A
50	D4	00031
	04	00033

```

ADDL3      #1, STRING, FAODSC+4
PUSHAB     P1
PUSHAB     DISDSC
PUSHAB     OUTRAB+34
PUSHL      R2
CALLS      #4, SYSS$FAOL
PUSHAB     OUTRAB
CALLS      #1, SYSS$PUT
CLRL       R0
RET

```

4803  
4807  
4809  
4812

; Routine Size: 52 bytes, Routine Base: \$CODES + 28F7

## help\_uaf - help routine

```
4756 4813 1 %sbttl 'help_uaf - help routine'
4757 4814 1 global routine help_uaf : novalue =
4758 4815 begin
4759 4816
4760 4817 ++
4761 4818
4762 4819 FUNCTIONAL DESCRIPTION:
4763 4820
4764 4821     Print out the help message or messages.
4765 4822
4766 4823 INPUTS:
4767 4824
4768 4825     none
4769 4826
4770 4827 OUTPUTS:
4771 4828
4772 4829     none
4773 4830
4774 4831 ROUTINE VALUE:
4775 4832
4776 4833     none
4777 4834
4778 4835 SIDE EFFECTS:
4779 4836
4780 4837     none
4781 4838 --
4782 4839
4783 4840 map
4784 4841     cmdlindsc: vector;
4785 4842
4786 4843 local
4787 4844     line_dsc      : vector [2];
4788 4845
4789 4846 line_dsc[0] = .cmdlindsc[0];
4790 4847 line_dsc[1] = .cmdlindsc[1];
4791 4848
4792 4849
4793 4850     The first thing to do is to remove 'help' from the command line
4794 4851     and give the help routine the remainder of the command line.
4795 4852     Find the beginning of the word 'help'.
4796 4853
4797 4854 if (.line_dsc[1])<0.8> eql ' '
4798 4855 then
4799 4856     begin
4800 4857         line_dsc[1] = ch$find_not_ch (.line_dsc[0], .line_dsc[1], ' ');
4801 4858         line_dsc[0] = .line_dsc[0] - (.line_dsc[1] - .cmdlindsc[1]);
4802 4859     end;
4803 4860
4804 4861
4805 4862     Now skip past the 'help' to the first blank (if there is one)
4806 4863
4807 4864 if ch$fail (line_dsc[1] = ch$find_ch (.line_dsc[0], .line_dsc[1], ' '))
4808 4865 then
4809 4866
4810 4867     No blank, set empty string
4811 4868
4812 4869     line_dsc[0] = 0
```

```
4813 2 else
4814 22
4815 22 Found a blank, set pointer to it
4816 22
4817 22 line_dsc[0] = .cmdlindsc[0] - (.line_dsc[1] - .cmdlindsc[1]);
4818 22
4819 22
4820 22 Start an interactive HELP session
4821 22
4822 22
4823 22 if not lbr$output_help (lib$put_output, 0, line_dsc,
4824 22 $descriptor ('uafhelp'), 0,
4825 22 lib$get_input)
4826 22 then LIB$SIGNAL(UAF$_HELPERR);
4827 22
4828 1 end;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

```
70 6C 65 68 66 61 75 0088C P.AFI: .ASCII \uafhelp\
                                00893 .BLKB 1
                                00000007 00894 P.AFH: .LONG 7
                                00000000 00898 .ADDRESS P.AFI
```

.PSECT \$CODE\$,NOWRT,2

				000C 00000	.ENTRY	HELP UAF, Save R2,R3	4814
	5E		04	C2 00002	SUBL2	#4, SP	
	53	00000000'	00	D0 00005	MOVL	CMDLINDSC, R3	4846
			53	DD 0000C	PUSHL	R3	
	52	00000000'	00	D0 0000E	MOVL	CMDLINDSC+4, R2	4847
04	AE		52	D0 00015	MOVL	R2, LINE_DSC+4	
	20	04	BE	91 00019	CMPB	@LINE_DSC+4, #32	4854
			15	12 0001D	BNEQ	2\$	
04	BE		20	3B 0001F	SKPC	#32, LINE_DSC, @LINE_DSC+4	4857
			02	12 00024	BNEQ	1\$	
			51	D4 00026	CLRL	R1	
	04	AE	51	D0 00028	1\$: MOVL	R1, LINE_DSC+4	
	50		52	C3 0002C	SUBL3	LINE_DSC+4, R2, R0	4858
		04	50	C0 00031	ADDL2	R0, LINE_DSC	
04	BE		20	3A 00034	2\$: LOCC	#32, LINE_DSC, @LINE_DSC+4	4864
			02	12 00039	BNEQ	3\$	
			51	D4 0003B	CLRL	R1	
	04	AE	51	D0 0003D	3\$: MOVL	R1, LINE_DSC+4	
			04	12 00041	BNEQ	4\$	
			6E	D4 00043	CLRL	LINE_DSC	4869
			09	11 00045	BRB	5\$	
	50		52	AE C3 00047	4\$: SUBL3	LINE_DSC+4, R2, R0	4874
	6E		50	C1 0004C	ADDL3	R3, R0, LINE_DSC	
		00000000G	00	9F 00050	5\$: PUSHAB	LIB\$GET_INPUT	4880
			7E	D4 00056	CLRL	-(SP)	
		00000000'	00	9F 00058	PUSHAB	P.AFH	4881
		0C	AE	9F 0005E	PUSHAB	LINE_DSC	4880



UAFMAIN  
V04-000

help\_uaf - help routine

C B  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1

Page 177  
(39)

		00000000G	7E	D4	00061	CLRL	-(SP)	
00000000G	00		00	9F	00063	PUSHAB	LIB\$PUT_OUTPUT	
	0D		06	FB	00069	CALLS	#6, LBR\$OUTPUT_HELP	
		00000000G	50	E8	00070	BLBS	R0, 6\$	
00000000G	00		8F	DD	00073	PUSHL	#UAF\$_HELPERR	4883
			01	FB	00079	CALLS	#1, LIB\$SIGNAL	
			04	00080	6\$:	RET		4885

; Routine Size: 129 bytes,    Routine Base: \$CODE\$ + 2928

exit\_uaf - normal exit routine

```
4830 4886 1 %sbttl 'exit_uaf - normal exit routine'
4831 4887 1 global routine exit_uaf : novalue =
4832 4888 begin
4833 4889
4834 4890 ++
4835 4891
4836 4892 FUNCTIONAL DESCRIPTION:
4837 4893
4838 4894 Normal exit routine.
4839 4895
4840 4896 INPUTS:
4841 4897
4842 4898 none
4843 4899
4844 4900 OUTPUTS:
4845 4901
4846 4902 none
4847 4903
4848 4904 ROUTINE VALUE:
4849 4905
4850 4906 none
4851 4907
4852 4908 SIDE EFFECTS:
4853 4909
4854 4910 image exits
4855 4911 --
4856 4912
4857 4913 %close (fab = uaffab);
4858 4914
4859 4915
4860 4916 Inform user of file modifications.
4861 4917
4862 4918
4863 4919 if .modify_flag
4864 4920 then
4865 4921
4866 4922 File has been modified
4867 4923
4868 4924 LIB$SIGNAL(UAF$_DONEMSG) ! tell user all is done
4869 4925 else
4870 4926
4871 4927 Here, no modifications were made to file.
4872 4928
4873 4929 LIB$SIGNAL(UAF$_NOMODS);
4874 4930
4875 4931 if .netuaf_exists
4876 4932 then
4877 4933 begin
4878 4934 if not .netuaf_modified
4879 4935 then
4880 4936 LIB$SIGNAL(UAF$_NAFNOMODS)
4881 4937 else
4882 4938 LIB$SIGNAL(UAF$_NAFDONEMSG);
4883 4939 end;
4884 4940
4885 4941 if .rdb_exists
4886 4942 then
```

exit\_uaf - normal exit routine

```
4887 4943 begin
4888 4944 if .rightslist_modified
4889 4945 then
4890 4946
4891 4947     File has been modified
4892 4948
4893 4949     LIB$SIGNAL(UAF$_RDBDONEMSG)      ! tell user all is done
4894 4950 else
4895 4951
4896 4952     Here, no modifications were made to file.
4897 4953
4898 4954     LIB$SIGNAL(UAF$_RDBNOMODS);
4899 4955 end ;
4900 4956
4901 4957 Sexit (code = true);
4902 4958 end;
```

```
00000000G 00 001C 00000
54 00000000' 00 9E 00002
53 00000000' 00 9E 00009
52 00000000G 00 9E 00010
00000000G 54 DD 00017
00 01 FB 00019
08 F154 C4 E9 00020
00000000G 8F DD 00025
06 11 0002B
00000000G 8F DD 0002D 1$:
62 01 FB 00033 2$:
16 63 E9 00036
08 F158 C4 E8 00039
00000000G 8F DD 0003E
06 11 00044
00000000G 8F DD 00046 3$:
62 01 FB 0004C 4$:
15 04 A3 E9 0004F 5$:
08 0C A3 E9 00053
00000000G 8F DD 00057
06 11 0005D
00000000G 8F DD 0005F 6$:
62 01 FB 00065 7$:
00000000G 00 01 DD 00068 8$:
01 FB 0006A
04 00071
```

.EXTRN SYS\$EXIT

```
.ENTRY EXIT UAF, Save R2,R3,R4
MOVAB UAF$AB, R4
MOVAB NETUAF_EXISTS, R3
MOVAB LIB$SIGNAL, R2
PUSHL R4
CALLS #1, SYS$CLOSE
BLBC MODIFY_FLAG, 1$
PUSHL #UAF$_DONEMSG
BRB 2$
PUSHL #UAF$_NOMODS
CALLS #1, LIB$SIGNAL
BLBC NETUAF_EXISTS, 5$
BLBS NETUAF_MODIFIED, 3$
PUSHL #UAF$_RANOMODS
BRB 4$
PUSHL #UAF$_RANFONEMSG
CALLS #1, LIB$SIGNAL
BLBC RDB_EXISTS, 8$
BLBC RIGHTS_LIST_MODIFIED, 6$
PUSHL #UAF$_RDBDONEMSG
BRB 7$
PUSHL #UAF$_RDBNOMODS
CALLS #1, LIB$SIGNAL
PUSHL #1
CALLS #1, SYS$EXIT
RET
```

```
4887
4913
4919
4924
4929
4931
4934
4936
4938
4941
4944
4949
4954
4957
4958
```

; Routine Size: 114 bytes, Routine Base: \$CODE\$ + 29AC

UAFMAIN  
V04-000

SIGNAL\_SYNTAX - Report missing qualifier

F 8  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 180  
(41)

```
: 4904      4959 1 %sbttl 'SIGNAL_SYNTAX - Report missing qualifier'
: 4905      4960 1 global routine SIGNAL_SYNTAX: novalue =
: 4906      4961 2 begin
: 4907      4962 2
: 4908      4963 2     LIB$SIGNAL(UAF$_ZISQUAL);
: 4909      4964 2
: 4910      4965 1 end;
```

```
00000000G 00 00000000G 8F DD 00002
01 FB 00008
04 0000F
```

```
.ENTRY SIGNAL_SYNTAX, Save nothing
PUSHL #UAF$_ZISQUAL
CALLS #1, LIB$SIGNAL
RET
```

```
: 4960
: 4963
:
: 4965
```

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 2A1E

acc\$exit - exit and cleanup routine

```

: 4912 4966 1 %sbttl 'acc$exit - exit and cleanup routine'
: 4913 4967 1 routine acc$exit : novalue =
: 4914 4968 2 begin
: 4915 4969 2
: 4916 4970 2 ++
: 4917 4971 2
: 4918 4972 2 FUNCTIONAL DESCRIPTION:
: 4919 4973 2
: 4920 4974 2 Exit on error condition.
: 4921 4975 2
: 4922 4976 2 INPUTS:
: 4923 4977 2
: 4924 4978 2 none
: 4925 4979 2
: 4926 4980 2 OUTPUTS:
: 4927 4981 2
: 4928 4982 2 none
: 4929 4983 2 --
: 4930 4984 2
: 4931 4985 2 $exit ();
: 4932 4986 2 end;

```

0000 00000 ACC\$EXIT:

		01 DD 00002	.WORD	Save nothing	: 4967
		01 FB 00004	PUSHL	#1	: 4985
00000000G 00		04 0000B	CALLS	#1, SYS\$EXIT	: 4986
			RET		

: Routine Size: 12 bytes, Routine Base: \$CODE\$ + 2A2E



```
4934 4987 1 %sbttl 'UAF$MOD SYS_PWD -- modify the system password'
4935 4988 1 global routine UAF$MOD_SYS_PWD: novalue =
4936 4989 begin
4937 4990
4938 4991 |++
4939 4992
4940 4993 Functional Description:
4941 4994
4942 4995 Modify the system password record by doing a $PUT on the UAF
4943 4996 with the UIF bit set for the username '<System+Password>'
4944 4997
4945 4998 |--
4946 4999
4947 5000 local
4948 5001
4949 5002 RBF,
4950 5003 PWD: block[DSC$K_D_BLN, byte]
4951 5004 preset([DSC$B_CLASS] = DSC$K_CLASS_D),
4952 5005 ROP;
4953 5006
4954 5007
4955 5008 Save current settings
4956 5009
4957 5010
4958 5011 RBF = .UAFRAB[RAB$R_RBF];
4959 5012 ROP = .UAFRAB[RAB$R_ROP];
4960 5013
4961 5014
4962 5015 Get new password and encrypt it
4963 5016
4964 5017
4965 5018 CLISGET_VALUE(%ascid'SYSTEM_PASSWORD', PWD);
4966 5019
4967 5020 if .PWD[DSC$W_LENGTH] neq 0 then
4968 5021 LGISHPWD(REC_ENCRYPT_DSC, PWD, UAF$C_PURDY_V, 0, %ASCID'<System+Password>')
4969 5022 else
4970 5023 ch$fill(0, UAF$S_PWD, RECBUF[UAF$Q_PWD]);
4971 5024
4972 5025
4973 5026 Fill in fields of the UAF record
4974 5027
4975 5028
4976 5029 ch$copy(17, UPLIT('<System+Password>'), ' ',
4977 5030 UAF$S_USERNAME, RECBUF[UAF$T_USERNAME]);
4978 5031 RECBUF[UAF$W_SALT] = 0;
4979 5032
4980 5033
4981 5034 Modify the RAB for our needs
4982 5035
4983 5036
4984 5037 UAFRAB[RAB$V_UIF] = 1;
4985 5038 UAFRAB[RAB$W_RSZ] = UAF$C_LENGTH;
4986 5039 UAFRAB[RAB$R_RBF] = RECBUF;
4987 5040
4988 5041
4989 5042 Modify the system password
4990 5043
```

```
4991 5044 2
4992 5045 2 $PUT(RAB=UAFRAB);
4993 5046 2
4994 5047 2
4995 5048 2
4996 5049 2
4997 5050 2
4998 5051 2 UAFRAB[RAB$$_RBF] = .RBF;
4999 5052 2 UAFRAB[RAB$$_ROP] = .ROP;
5000 5053 2
5001 5054 1 end;
```

```
44 52 4F 57 53 53 41 50 5F 4D 45 54 53 59 53 0089C P.AFK: .PSECT $SPLITS,NOWRT,NOEXE,2
52 00 008AB .ASCII \SYSTEM_PASSWORD\<0>
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008B0 P.AFJ: .LONG 17694735
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008B4 P.AFM: .ADDRESS P.AFK
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008C3 P.AFM: .ASCII \<System+Password>\<0><0><0>
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008C8 P.AFL: .LONG 17694737
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008CC P.AFL: .ADDRESS P.AFM
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008D0 P.AFN: .ASCII \<System+Password>\<0><0><0>
72 6F 77 73 73 61 50 2B 6D 65 74 73 79 53 3C 008DF
```

```
03FC 00000
59 00000000 00 9E 00002
58 00000000 00 9E 00009
5E 02000000 04 C2 00010
04 8F DD 00013
57 04 AE D4 00019
56 DC A8 D0 0001C
00000000G 00 8F BB 0001F
00 02 FB 00023
00 6E B5 00027
00 16 13 0002E
00 1C A9 9F 00030
7E 02 7D 00032
0C AE 9F 00035
F940 C9 9F 00038
00000000G 00 05 FB 0003B
00 08 11 0003F
08 00 2C 00046
20 00 2C 00048 18:
00 FB44 C8 0004D
00 11 2C 00050 28:
00 F9F4 C8 00056
00 FB56 C8 B4 00059
00 DC A8 10 88 0005D
00 FA AB 0584 8F B0 00061
```

```
.PSECT $CODES,NOWRT,2
.ENTRY UAF$MOD_SYS_PWD, Save R2,R3,R4,R5,R6,R7,R8,-, 4988
R9
MOVAB P.AFJ, R9
MOVAB UAFRAB+40, R8
SUBL2 #4, SP
PUSHL #33554432
CLRL PWD+4
MOVL UAFRAB+40, RBF
MOVL UAFRAB+4, ROP
PUSHR #M<R9, SP>
CALLS #2, CLISGET_VALUE
TSTW PWD
BEQL 18
PUSHAB P.AFL
MOVQ #2, -(SP)
PUSHAB PWD
PUSHAB REC_ENCRYPT_DSC
CALLS #5, LGISHPWD
BRB 28
MOVC5 #0, (SP), #0, #8, RECBUF+340
MOVC5 #17, P.AFN, #32, #32, RECBUF+4
CLRQ RECBUF+358
BISB2 #16, UAFRAB+4
MOVW #1412, UAFRAB+34
```

```
5004
5011
5012
5018
5020
5021
5023
5030
5031
5037
5038
```

UAFMAIN  
V04-000

UAF\$MOD\_SYS\_PWD -- modify the system password

J 8  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1  
Page 184  
(43)

	68	F9F0	C8	9E	00067
		D8	A8	9F	0006C
00000000G	00		01	FB	0006F
	68		57	D0	00076
DC	A8		56	D0	00079
			04	0007D	

MOVAB	RECBUF, UAFRAB+40
PUSHAB	UAFRAB
CALLS	#1, SYSSPUT
MOVL	RBF, UAFRAB+40
MOVL	ROP, UAFRAB+4
RET	

:	5039
:	5045
:	
:	5051
:	5052
:	5054

; Routine Size: 126 bytes,    Routine Base: \$CODE\$ + 2A3A

security\_audit - perform a security audit

```
5003 5055 1 %sbttl 'security_audit - perform a security audit'
5004 5056 1 routine security_audit (code, old_user) : novalue =
5005 5057 begin
5006 5058
5007 5059 ++
5008 5060
5009 5061 FUNCTIONAL DESCRIPTION:
5010 5062
5011 5063     Perform a security audit, if needed.
5012 5064
5013 5065 INPUTS:
5014 5066
5015 5067     CODE - Security audit record id
5016 5068     OLD_USER - Old username (optional depending on the record id)
5017 5069
5018 5070 IMPLICIT INPUTS:
5019 5071
5020 5072     PCB_STS - This process's PCB status
5021 5073     UAF$GQ_SYSUAFF - SYSUAF fields modified
5022 5074
5023 5075 OUTPUTS:
5024 5076
5025 5077     none
5026 5078
5027 5079 IMPLICIT OUTPUTS:
5028 5080
5029 5081     none
5030 5082
5031 5083 ROUTINE VALUE:
5032 5084
5033 5085     none
5034 5086
5035 5087 SIDE EFFECTS:
5036 5088
5037 5089     none
5038 5090
5039 5091 --
5040 5092
5041 5093 external routine
5042 5094     nsa$event_audit;                ! Kernel mode auditing routine
5043 5095
5044 5096 external
5045 5097     nsa$gr_alarmvec : block [,byte], ! Security audit alarm vector
5046 5098     nsa$gr_journvec : block [,byte]; ! Security audit journal vector
5047 5099
5048 5100 macro
5049 5101     add_quad_packet(type, a_quad) = ! Add a quadword packet to list
5050 5102     begin
5051 5103         (.arglist_ptr)<0,16> = %name('nsa$%k_pkttyp_', type);
5052 5104         arglist_ptr = .arglist_ptr + 2;
5053 5105         (.arglist_ptr)<0,16> = nsa$%k_arg_mech_quad;
5054 5106         arglist_ptr = .arglist_ptr + 2;
5055 5107         (.arglist_ptr)<0,32> = (.a_quad)<0,32>;
5056 5108         arglist_ptr = .arglist_ptr + 4;
5057 5109         (.arglist_ptr)<0,32> = (.a_quad)<32,32>;
5058 5110         arglist_ptr = .arglist_ptr + 4;
5059 5111         arglist[nsa$b_arg_pktnum] = .arglist[nsa$b_arg_pktnum] + 1;
```

```
5060      end%  
5061      add_descr_packet(type, len, ptr) = ! Add a descriptor packet to list  
5062      begin  
5063          (.arglist_ptr)<0,16> = %name('nsa$pk_pkttyp_', type);  
5064          arglist_ptr = .arglist_ptr + 2;  
5065          (.arglist_ptr)<0,16> = nsa$arg_mech_descr;  
5066          arglist_ptr = .arglist_ptr + 2;  
5067          (.arglist_ptr)<0,32> = len;  
5068          arglist_ptr = .arglist_ptr + 4;  
5069          (.arglist_ptr)<0,32> = ptr;  
5070          arglist_ptr = .arglist_ptr + 4;  
5071          arglist[nsa$b_arg_pktnum] = .arglist[nsa$b_arg_pktnum] + 1;  
5072      end%  
5073  
5074      local  
5075          arglist : block ! Argument list for NSA$EVENT_AUDIT  
5076          [nsa$arg_hdr_length + ((2 + 2 + 8) * 6), byte],  
5077          arglist_ptr; ! Packet fill in pointer  
5078  
5079      ch$fill(0, %allocation(arglist), arglist); ! Clear out the argument list  
5080  
5081      if .nsa$gr_alarmvec[nsa$v_evt_uaf] ! If alarm auditing,  
5082      then arglist[nsa$v_arg_flag_alarm] = true; ! perform alarm audit  
5083  
5084      if .nsa$gr_journvec[nsa$v_evt_uaf] ! If journal auditing,  
5085      then arglist[nsa$v_arg_flag_journ] = true; ! perform journal audit  
5086  
5087      if .pcb_sts[%bitposition(pcb$v_secaudit)] ! If mandatory auditing,  
5088      then arglist[nsa$v_arg_flag_mandy] = true; ! perform mandatory audit  
5089  
5090      if .arglist[nsa$b_arg_flag] eql 0 ! If no audit requested,  
5091      then return; ! simply exit  
5092  
5093      arglist[nsa$l_arg_id] = .code; ! Set audit record type/subtype  
5094  
5095      arglist_ptr = arglist[nsa$t_arg_list]; ! Address packet(s) in arg list  
5096  
5097      if .arglist[nsa$w_arg_type] eql nsa$pk_rectyp_sysuaf  
5098      then ! For SYSUAF records...  
5099          begin  
5100              if .arglist[nsa$w_arg_subtype] neq nsa$pk_rectyp_sysuaf_del  
5101              then  
5102                  add_quad_packet(sysuaff, uaf$gq_sysuaff);  
5103                  add_descr_packet(username,  
5104                      namelen(uaf$s_username, recbuf[uaf$t_username]),  
5105                      recbuf[uaf$t_username]);  
5106                  if .arglist[nsa$w_arg_subtype] eql nsa$pk_rectyp_sysuaf_cop  
5107                  or .arglist[nsa$w_arg_subtype] eql nsa$pk_rectyp_sysuaf_ren  
5108                  then  
5109                      add_descr_packet(username,  
5110                          namelen(uaf$s_username, .old_user),  
5111                          .old_user);  
5112                  end  
5113              else ! For NETUAF records...  
5114                  begin  
5115                      add_descr_packet(nodenam,  
5116                          namelen(naf$s_node, netbuf[naf$t_node]),
```



```
5117      netbuf[naf$t_node]);
5118      add_descr_packet(username,
5119      namelen(naf$s_remuser, netbuf[naf$t_remuser]),
5120      netbuf[naf$t_remuser]);
5121      add_descr_packet(username,
5122      namelen(naf$s_localuser, netbuf[naf$t_localuser]),
5123      netbuf[naf$t_localuser]);
5124      if .arglist[nsa$w_arg_subtype] eql nsa$k_rectyp_netuaf_mod
5125      then
5126      begin
5127      add_descr_packet(nodenam,
5128      namelen(naf$s_node, netbuf[naf$t_node]),
5129      netbuf[naf$t_node]);
5130      add_descr_packet(username,
5131      namelen(naf$s_remuser, netbuf[naf$t_remuser]),
5132      netbuf[naf$t_remuser]);
5133      add_descr_packet(username,
5134      namelen(uaf$s_username, .old_user),
5135      .old_user);
5136      end;
5137      end;
5138      arglist[nsa$l_arg_count] = (.arglist_ptr - (arglist + 4)) / 4; ! Set # args
5139      $cmkrnl(routin = nsa$event_audit, arglist = arglist); ! Do the security audit
5140
5141
5142
5143      end;
```

```
.EXTRN  NSASEVENT_AUDIT
.EXTRN  NSASGR_ALARMVEC
.EXTRN  NSASGR_JOURNVEC
.EXTRN  SYSSCMRNL
```

## 007C 00000 SECURITY\_AUDIT:

0054	8F	00	56	00000000	00	9E	00002	WORD	Save R2,R3,R4,R5,R6	5056
			5E	AC	AE	9E	00009	MOVAB	NETBUF, R6	
			6E		00	2C	0000D	MOVAB	-84(SP), SP	
					6E		00014	MOVCS	#0, (SP), #0, #84, ARGLIST	5131
		04	00000000G	00	02	E1	00015	BBC	#2, NSASGR_ALARMVEC, 1\$	5133
			08	AE	01	88	0001D	BISB2	#1, ARGLIST+8	5134
		04	00000000G	00	02	E1	00021	BBC	#2, NSASGR_JOURNVEC, 2\$	5136
			08	AE	02	88	00029	BISB2	#2, ARGLIST+8	5137
		04	00000000	00	03	E1	0002D	BBC	#3, PCB_STS+3, 3\$	5139
			08	AE	04	88	00035	BISB2	#4, ARGLIST+8	5140
				08	AE	95	00039	TSTB	ARGLIST+8	5142
					01	12	0003C	BNEQ	4\$	
						04	0003E	RET		
		04		AE	04	AC	0003F	MOVL	CODE, ARGLIST+4	5145
			52	0C	AE	9E	00044	MOVAB	ARGLIST+12, ARGLIST_PTR	5147
			02	04	AE	B1	00048	CMPL	ARGLIST+4, #2	5149
					3C	12	0004C	BNEQ	6\$	
			02	06	AE	B1	0004E	CMPL	ARGLIST+6, #2	5152
					0F	13	00052	BEQL	5\$	
		82	00030012	BF	D0	00054	MOVL	#196626, (ARGLIST_PTR)+	5154	

		82	F9F8	C6	7D	0005B	MOVQ	UAF\$GQ_SYSUAFF, (ARGLIST_PTR)+		
			09	AE	96	00060	INCB	ARGLIST+9		
FA80	C6	82	0004000A	8F	D0	00063	5\$:	MOVL	#262154, (ARGLIST_PTR)+	5157
	82	20		20	3A	0006A	LOCC	#32, #32, RECBUF+4		
		20		50	C3	00070	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82	FA80	C6	9E	00074	MOVAB	RECBUF+4, (ARGLIST_PTR)+		
			09	AE	96	00079	INCB	ARGLIST+9		
		04	06	AE	B1	0007C	CMPW	ARGLIST+6, #4	5158	
				7D	13	00080	BEQL	8\$		
		05	06	AE	B1	00082	CMPW	ARGLIST+6, #5	5159	
				49	12	00086	BNEQ	7\$		
				75	11	00088	BRB	8\$	5163	
		82	00040009	8F	D0	0008A	6\$:	MOVL	#262153, (ARGLIST_PTR)+	5169
	66	20		20	3A	00091	LOCC	#32, #32, NETBUF		
	82	20		50	C3	00095	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82		66	9E	00099	MOVAB	NETBUF, (ARGLIST_PTR)+		
			09	AE	96	0009C	INCB	ARGLIST+9		
		82	0004000A	8F	D0	0009F	MOVL	#262154, (ARGLIST_PTR)+	5172	
20	A6	20		20	3A	000A6	LOCC	#32, #32, NETBUF+32		
	82	20		50	C3	000AB	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82	20	A6	9E	000AF	MOVAB	NETBUF+32, (ARGLIST_PTR)+		
			09	AE	96	000B3	INCB	ARGLIST+9		
		82	0004000A	8F	D0	000B6	MOVL	#262154, (ARGLIST_PTR)+	5175	
40	A6	20		20	3A	000BD	LOCC	#32, #32, NETBUF+64		
	82	20		50	C3	000C2	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82	40	A6	9E	000C6	MOVAB	NETBUF+64, (ARGLIST_PTR)+		
			09	AE	96	000CA	INCB	ARGLIST+9		
		03	06	AE	B1	000CD	CMPW	ARGLIST+6, #3	5176	
				43	12	000D1	7\$:	BNEQ	9\$	
		82	00040009	8F	D0	000D3	MOVL	#262153, (ARGLIST_PTR)+	5181	
	66	20		20	3A	000DA	LOCC	#32, #32, NETBUF		
	82	20		50	C3	000DE	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82		66	9E	000E2	MOVAB	NETBUF, (ARGLIST_PTR)+		
			09	AE	96	000E5	INCB	ARGLIST+9		
		82	0004000A	8F	D0	000E8	MOVL	#262154, (ARGLIST_PTR)+	5184	
20	A6	20		20	3A	000EF	LOCC	#32, #32, NETBUF+32		
	82	20		50	C3	000F4	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82	20	A6	9E	000F8	MOVAB	NETBUF+32, (ARGLIST_PTR)+		
			09	AE	96	000FC	INCB	ARGLIST+9		
		82	0004000A	8F	D0	000FF	8\$:	MOVL	#262154, (ARGLIST_PTR)+	5187
08	BC	20		20	3A	00106	LOCC	#32, #32, @OLD USER		
	82	20		50	C3	0010B	SUBL3	R0, #32, (ARGLIST_PTR)+		
		82	08	AC	D0	0010F	MOVL	OLD USER, (ARGLIST_PTR)+		
			09	AE	96	00113	INCB	ARGLIST+9		
		50	04	AE	9E	00116	9\$:	MOVAB	ARGLIST+4, R0	5191
		52		50	C2	0011A	SUBL2	R0, R2		
6E		52		04	C7	0011D	DIVL3	#4, R2, ARGLIST		
				5E	DD	00121	PUSHL	SP	5193	
			00000000G	00	9F	00123	PUSHAB	NSA\$EVENT AUDIT		
				02	FB	00129	CALLS	#2, SYSS\$CMKRN	5195	
				04	00	00130	RET			

; Routine Size: 305 bytes, Routine Base: \$CODE\$ + 2AB8

UAFMAIN  
V04-000

security\_audit - perform a security audit

: 5145  
: 5146

5196 1 end  
5197 0 eludom

B 9  
16-Sep-1984 02:16:54  
14-Sep-1984 13:21:22

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[UAF.SRC]UAFMAIN.B32;1 (45)

Page 189

! End of module

.EXTRN LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$SPLITS	2276	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS	4364	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$GLOBALS	2032	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	11241	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
. ABS .	0	NOVEC,NOWRT,NORD ,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	300	1	1000	00:02.0

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:UAFMAIN/CBJ=OBJ\$:UAFMAIN MSRC\$:UAFMAIN/UPDATE=(ENH\$:UAFMAIN)

; Size: 11241 code + 8672 data bytes  
; Run Time: 03:09.3  
; Elapsed Time: 04:20.7  
; Lines/CPU Min: 1646  
; Lexemes/CPU-Min: 26564  
; Memory Used: 654 pages  
; Compilation Complete



0406 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

RIGHTSMAN  
LTS

DAFMATN  
LTS



0407 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

0407	0408	0409	0410	0411	0412	0413	0414	0415	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763	1764	1765	1766	1767	1768
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------